ISSN 0998-9250 Mars 1991 / N° 11

Avec cahier Dr Dobb's Journal

DEBUG magazine

La Face Cachée de Bill Gates

une interview exclusive
par JOHN PERRY BARLOW

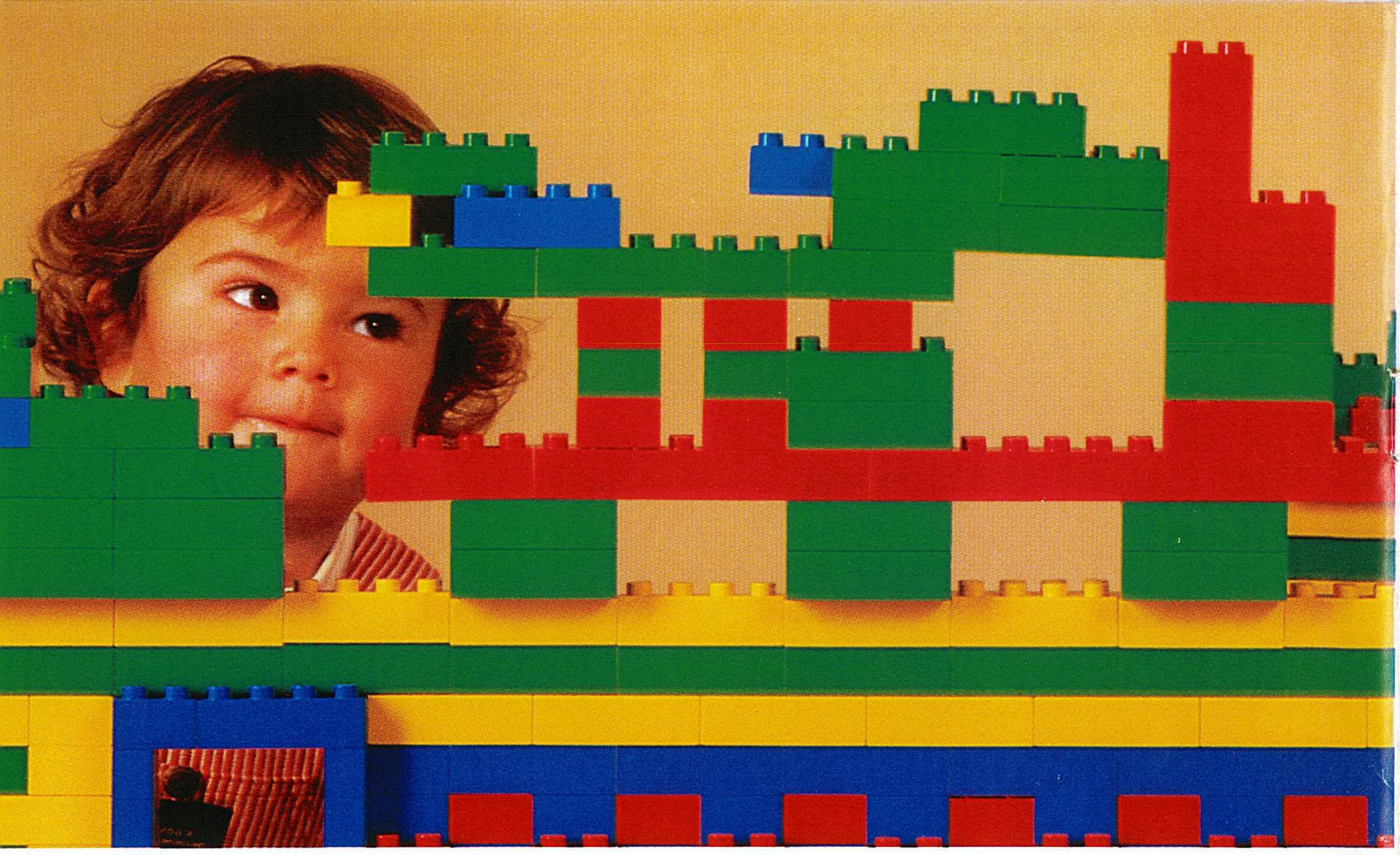
« Vous sentez-vous investi d'une mission ? »
« Oui »

« Avez-vous toujours la possibilité d'exercer vos talents ? »
« Tout à fait. En matière de définition de produits, d'algorithmes... C'est même ce que je préfère à toute autre chose ».

La Force Est Toujours Avec Lui...

M 3032 - 11 - 37,00 F

Belgique : 258 FB - Suisse : 11,5 FS



ObjectVision de Borland maintenant, tout le monde peut créer des applications professionnelles pour Windows

Il n'y a jamais rien eu de comparable à ObjectVision. Pour la première fois, des utilisateurs sans connaissance technique particulière, vont pouvoir concevoir et mettre à jour rapidement et facilement leurs propres applications.

Imaginez les avantages !

- Développer de nouveaux programmes Windows et les utiliser rapidement.
- Créer facilement des interfaces pour des bases de données.
- Se connecter aux bases de données de l'entreprise et à d'autres applications Windows.
- Réduire le temps nécessaire à la formation des utilisateurs.
- Eliminer les lourdeurs de la programmation.

La Programmation Visuelle

Avec ObjectVision, Borland introduit un concept révolutionnaire dans les logiciels de programmation visuelle. Désormais, si vous êtes en mesure de décrire votre application, vous pouvez la créer vous-même.

1. Décidez à quoi doit ressembler votre application.
2. Dessinez l'arbre de décision (ou l'organigramme) qui détermine la logique de votre application.

Ajoutez des formules de calcul de type tableur ou reliez votre programme à d'autres applications Windows, ou à d'autres bases de données.

Offre Exceptionnelle limitée dans le temps *

Avec un prix de lancement de 995 F HT seulement (au lieu de 4995 F HT), c'est le moment ou jamais de vous intéresser à ObjectVision.

Adressez-vous à votre revendeur habituel ou téléphonez dès maintenant au N° VERT 05.46.96.69.

APPEL GRATUIT

ObjectVision est également disponible en version Runtime. Cela vous permet de diffuser vos applications ObjectVision auprès d'autres utilisateurs.

* Offre exceptionnelle valable jusqu'au 30 juin 1991 chez votre revendeur habituel.

BORLAND

Editeur de Paradox®, Quattro® Pro, Sprint®, Turbo C++ et Turbo Pascal®

43, avenue de l'Europe – BP 6 78143 Vélizy-Villacoublay – France – Tél. (33) (1) 39.46.96.69 Télécopie (33) (1) 39.46.81.60 – Télex 698 793

BEAC

BAC A SABLE

EDITO

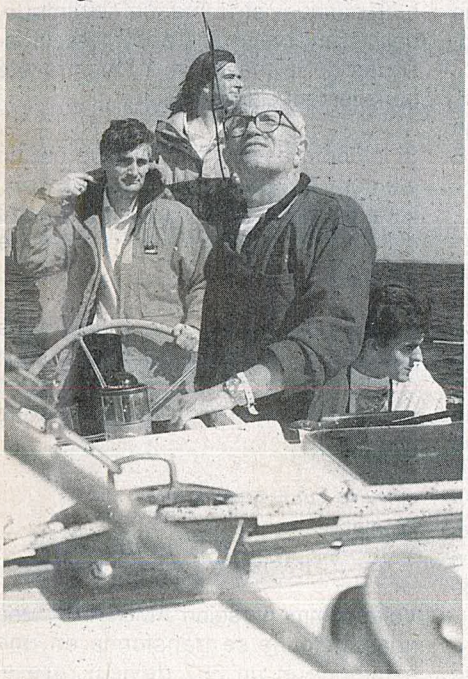
Pourquoi Microsoft a-t-il conçu OS/2 tel qu'il est? La presse, pour une fois (??!!), le déconseille.

Les bruits courent que la direction d'IBM, toujours prête à saisir une occasion pour remettre ses joujoux sur le marché, pousse OS/2 bien que le produit soit cher et, surtout, sans grande utilité. Le problème est que les utilisateurs forment aujourd'hui un groupe trop diversifié pour marcher au pas de Big Blue.

Que la société IBM soit ou non responsable de cette stratégie, n'explique pas pourquoi OS/2 ne répond pas aux besoins des utilisateurs. Chuchoter IBM tout en créant un système d'exploitation qui répondrait aux besoins des utilisateurs, aurait été un jeu d'enfants pour Microsoft.

Que veut, en réalité, l'utilisateur? Quelque chose de simple et au plus bas prix possible. Il ne veut pas payer des fonctions dont il ne ressent pas le besoin. Il ne veut pas investir du temps dans un apprentissage long et fastidieux. Il veut que cela marche du premier coup. Et que cela dure!

OS/2 échoue sur tous ces points. Le produit est cher à l'acquisition et à l'utilisation. Il y a un grand doute sur la pérennité de Presentation Manager. Il est compliqué à installer et à utiliser. Et comme dans toute innovation, les bugs sont monnaie courante. En conséquence, toute migration massive vers OS/2 est peu probable.



Elle exigerait des investissements énormes en logiciel et en matériel ainsi qu'un long apprentissage de sa nouvelle interface. Un combat dans un camp de bugs. Et ce joujou nous obligerait à payer à Microsoft des redevances périodiques, dissimulées sous une forme des mises à jour, au fur et à mesure que tous ces Mossieurs daignent corriger leurs erreurs.

J'ai lu avec attention le livre de Gordon Letwin - « Inside OS/2 » et je crois savoir pourquoi OS/2 échoue. Monsieur Letwin est architecte en chef du projet OS/2, ses compétences techniques sont au dessus de tout soupçon (rappelez vous HDOS?). Et c'est là que réside l'origine du problème.

Dans son livre, Gordon tente non seulement de décrire OS/2, il essaye aussi de nous dire pourquoi il est conçu ainsi. Nulle part on ne trouve l'indice d'une compréhension du marché auquel il s'adresse. OS/2 est un produit crée en labo, par le labo et pour le labo. Il existe un nom pour de tels produits, fruits de la démarche qui ne tiennent pas compte du marché auquel ils sont adressés: l'ingénierie du « BAC A SABLE ». L'expression évoque l'image d'enfants édifiant des châteaux dans un bac à sable. On leur autorise cela, car l'endroit s'y prête. Tant qu'ils y sont, peu importe ce qu'ils y fabriquent, ils ne dérangent personne.

Plus précisément, l'ingénierie du BAC A SABLE pointe son nez dés qu'on peut observer deux indices:

- l'ingénieur sait mieux que l'utilisateur ce qui est bon pour ce dernier - si l'on peut le faire, il faut le faire, car la valeur absolue d'un système est la somme globale de toutes ses fonctionnalités.

D'une manière générale, les ingénieurs sont des gens raisonnables. Le problème ne se situe pas au niveau de leur bon sens. Ils sont prisonniers de la culture laborantine. Ils sont isolés du monde des utilisateurs, à qui s'adressent leur produits. C'est cela qui les amène à concevoir des produits s'imposant à la force de la répétition. Et à la longue, cette répétition les renforce dans leurs convictions, même les plus absurdes. Si seulement ces idiots essayaient d'utiliser les fruits de leur créativité, ils se seraient rendu rapidement compte de leur bevue.

Ce que nous voulons, Monsieur Letwin, ce sont des produits peu chers, simples et performants. Des produits qui marchent du premier coup et qui satisfassent nos besoins réels.D'accord, c'est beaucoup plus que ce qu'exigeait de vous IBM ou Microsoft. Mais voyez le temps que nous perdons pendant que vous et vos copains jouez dans le bac à sable.

Alexeï Hoffman

PARLONS - EN

☞ ***« Notre Association, « Enfance et Education 95 », a décidée de renouveler son matériel informatique. Nous nous sommes penchés sur la nouvelle génération AMSTRAD PC. Suite à la documentation commerciale des spécifications du PC 3286, comprenant les caractéristiques de l'ordinateur, nous avons donc acheté le PC ainsi que ses options.***

Une fois le matériel configuré, nous nous sommes aperçus que l'ordinateur n'était pas équipé d'un coprocesseur 80287, comme nous a fait croire la documentation AMSTRAD. Nous nous retrouvons donc en possession d'un matériel incomplet.

AMSTRAD considère que cette erreur est une « ligne qui a sauté à l'impression » et nous conseille de se rendre chez le distributeur pour l'achat du coprocesseur ou remboursement de l'ensemble.

Voila maintenant une nouveauté qui laisse songeur tout utilisateur d'AMSTRAD. Je pense que AMSTRAD devrait respecter ses obligations contractuelles et fournir des matériels conformes à ses publicités. Vouloir s'informatiser donne le droit de demander au fournisseur une documentation technique et commerciale complète, sans erreurs, des conseils et mises en garde. En clair, éviter qu'une erreur « d'impression » ne devienne pour l'utilisateur un préjudice, malgré un erratum fantôme pour un client ».

Akim ROUICHI
Garges les Gonesse

Cette lettre nous est arrivée accompagnée de copies d'une volumineuse correspondance entre Mr Rouichi et la société AMSTRAD. Le premier document est le prospectus publicitaire AMSTRAD, décrivant le « PC GENERATION 3 », PC3286. Les spécifications techniques indiquent clairement l'existence d'un 80287 à cote d'un 80286, comme la vitesse d'horloge 16 MHz, type de bus asynchrone, batterie (10 ans), lecteur 3"1/2, disque dur 40Mo, VGA etc. Tout laisse croire que le coprocesseur est fourni au standard, inclus dans le prix.

La lettre de Mr Rouichi à Mme Marion Vanier, PDG d'AMSTRAD, datée du 15 novembre 1990, suit.
Mr Rouichi écrit notamment:

« (...) Sur présentation de votre documentation technique, j'ai décidé d'acquérir le PC3286, équipé d'un disque dur et ses options non fournies (souris, support document...). Une fois notre matériel mis en route ainsi que nos applications, nous nous sommes aperçus que la documentation descriptive du PC3286 ne correspondait pas à la configuration interne de la machine. Effectivement, l'ordinateur en notre possession n'est en aucun cas équipé d'un coprocesseur 80287.

De ce fait, j'ai immédiatement contacté vos services consommateurs et marketing. J'ai été renvoyé d'un service en service sans avoir d'explications concrètes sur la désinformation publicitaire concernant le produit cité et les démarches à suivre. Seule une personne du service juridique a été capable de répondre à mon attente. (...). Suite à l'affirmation d'une secrétaire qu'une note a été envoyée aux différents distributeurs, j'affirme que j'ai moi-même contacté le responsable informatique pour vérifier ce qui a été communiqué. Il s'avère qu'aucune note n'a pas été distribuée. Je tiens à vous signaler que j'ai occupé le poste de vendeur micro-informatique dans cette entreprise ».

Mme Vannier n'a pas répondu a cette lettre. Elle a confié cette tâche à son service marketing qui a fabriqué la réponse suivante (datée du 27/11/91):
« Nous accusons réception de votre courrier concernant l'existence d'un coprocesseur arithmétique dans les PC3286. Nous avons effectivement constaté qu'une erreur d'impression s'est glissée dans la première édition (tirage restreint) de notre documentation commerciale, document par ailleurs non contractuel. A la suite de quoi nous avons pris les mesures suivantes :
- réédition corrigée de la documentation
-erratum paru dans la revue AMSTRAD PC
- circulaire envoyée à tous nos distributeurs les informant de cette erreur. De plus, la campagne publicitaire presse et radio n'a jamais mentionné la présence la présence d'un coprocesseur arithmétique, et le manuel technique, définissant précisément les caractéristiques de la machine, a toujours été parfaitement clair à ce sujet.

Si le coprocesseur arithmétique est un élément déterminant de votre choix, vous pouvez l'acheter auprès de votre distributeur, qui pourra se charger de son installation.

A titre indicatif, nous vous informons que les PC équipés en standard d'un coprocesseur arithmétique sont rares sur le marché et d'un prix beaucoup plus élevé.
(...)
signé le service marketing
Nicolai ».

Voilà comment, selon AMSTRAD, une non-existence se transforme en une existence et un 287 devient rare et cher (à titre d'information nous pouvons indiquer à AMSTRAD quelques adresses pour acheter des 287 à 500 Frs, en quantité unitaire). Nous ne savons pas seulement si les méthodes de marketing d'AMSTRAD vont être acceptées par tous les clients. Qu'en pense Mr Rouichi, pris au dépourvu par cette brillante réthorique d'AMSTRAD ? Le voila qui saisit sa la plus belle plume, pour leur adresser la missive suivante (datée du 13 décembre 1990) : « A l'attention de Madame NETZDORF,

Suite à notre conversation téléphonique du 5 décembre 1990, (...) je pense que (...) un client qui achète un ordinateur suite à une fiche commerciale descriptive doit se retrouver logiquement avec le matériel cité sans aucun oubli (...). En aucun cas une

[SERVICE LECTEURS : INTERESSANT = 1 / MOYEN = 2/ MAUVAIS = 3]

DEBUG MAGAZINE

Rédacteur en Chef
Directeur de Publication
Artur G. Piotrowski

Maquettiste
Elisa Le Moal

Ont collaboré à ce numéro
ART, Jeff Duntemann, Hugh Mc Larty, David Smith, K.N. King, Philippe Dalléas, Pierre Morel Fourrier, Georges Bourne, Brian R. Anderson John Perry Barlow, Jean Guibert, E.M. Amblard. G. Korwin. Serge Grun. John C. Dvorak

Iconographie
Photos : Debug Magazine
Dessin : Pierre Marmoud

REDACTION
TELEMATIQUE
VENTE AU NUMERO
SERVICE LECTURE
GESTION DES ABONNEMENTS
EDITEUR
SOFTMEDIA
SARL de presse au c^{al} de 5000 F.
3, rue de l'Atlas
75019 Paris
Téléphone : (1) 42 02 29 16
Télécopie : (1) 42 45 58 13

Coordination Production
Christian Hernault

COMPTABILITE - ABONNEMENTS
Lidia Zakan

Prix de l'abonnement
11 numéros + 1 numéro spécial
486 F. T.T.C. (France)
Prix au numéro : 37 F.

IMPRIMERIE, PHOTOGRAVURE
GC Conseil

Mise en page : X.PRESS-MAC

Dépôt légal à la parution
Commission paritaire 71290
ISSN 0998-9250
membre inscrit à O.J.D.

« La loi du 11 Mars 1957 n'autorisant pas aux termes des alinéas 2 et 3 de l'article 41 d'une part que les copies ou reproductions strictement réservées à l'usage privée du copiste et non destinés à une utilisation collective et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, faites sans le consentement de son auteur ou de ses ayant droit ou ayant cause, est illicite (alinéa premier de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du code pénal.

SOMMAIRE

N° 11 - Mars 1991

EDITO 3

RUMEURS 10

DOSSIER
La Face Cachée de Bill Gates 12

DOSSIER Dr. DOBB'S
De Pascal à C++ 21
Une introduction à Modula 2 pour les programmeurs Pascal 27
ADA pour les programmeurs en Pascal 51
Kermit pour O/S 2 62

PROGRAMMATION
Cahier Listing 33

EVALUATION
Turbo Pascal 6.0, peut-on faire mieux ? 66

PODIUM : 72
IDEE : 75
ANNONCES 78

LA PAGE DE JOHN C. DVORAK 76

Ce numéro comporte un catalogue publicitaire encarté au centre, détachable.

documentation publicitaire (...) ne doit comporter d'erreurs pouvant jouer sur l'achat (...).

Prendre les mesures pour corriger les erreurs est effectivement une obligation mais :

- les consommateurs ne lisent pas tous AMSTRAD PC
- les consommateurs ne sont pas toute la journée à l'écoute de la radio
- les consommateurs ne sont pas responsables de la documentation déposée chez les distributeurs
- les consommateurs ne peuvent rien tirer de la nouvelle documentation (...)

Et les consommateurs ne peuvent pas, dans la plupart de cas, ouvrir un carton dans un magazine pour confirmer l'exactitude d'une publicité fournie (...).

Je trouve obsolète une structure qui vous renvoie de poste en poste sans fournir de réponses concrètes aux clients qui représentent votre chiffre d'affaire. Il est navrant de résoudre de simples litiges avec des personnes arrogantes possédant un faible potentiel de communication et aucune maîtrise face à un problème spécifique de l'AMSTRAD (...). Je pense que Madame Marion Vanier devrait lire le courrier qui lui est destiné afin d'éviter d'aboutir à la rupture avec de fidèles clients. Cela ne pourrait qu'engendrer une meilleure communication et Madame Nicolai serait déchargée d'une tâche stressante qui pourrait l'aider à retrouver une réelle aptitude au marketing communicatif. (...)

Quant à la plainte (concernant) le coprocesseur absent, suite à cette erreur plus que grossière, (...) il faudrait éviter de renvoyer le client chez le distributeur et vice versa; sachant qu'un fax n'est pas une preuve de dégagement de responsabilité. (...)

J'espère que ce courrier vous aidera à comprendre que ce n'est pas un coprocesseur qui détermine un choix mais un système fiable qui considère les clients d'AMSTRAD avant tout comme des clients. (...) ».

Que va répondre à ce message désespéré d'un client mécontent le « service marketing » de la société AMSTRAD? Rappelons qu' AMSTRAD réalise en France un chiffre d'affaire de plusieurs centaines de millions de francs. Va-t-il fournir gracieusement un coprocesseur qui coûte quelque 500 Frs à cette Association, sa cliente ? Va-t-il s'excuser platement et joindre une bouteille de champagne à l'attention de son responsable et acheteur ? Non. Même les petites économies sont bonnes à faire et un sou est un sou ! D'ailleurs, au service de marketing on emploie un régiment de personnes hautement qualifiées et payées pour évacuer ce genre de problèmes. Madame Nicolai se charge d'une réponse (lettre datée du 14/12/90) suivante :

« (...) si le coprocesseur a été un élément déterminant de votre achat, il vous appartient de demander à votre vendeur l'annulation de la vente et le remboursement du prix du PC contre restitution complète du matériel. Nous attirons néanmoins votre attention sur le fait que cette reprise du matériel ne pourra intervenir après le 31/12/90. Souhaitant avoir répondu à votre demande, Monsieur, en nos meilleures salutations».

Et toc! Qu'une vis ne manque, sinon gare à vous! Il vous reste deux semaines. Dépêchez-vous, ramassez vos cartons, manuels, câbles, effacez le disque dur, emballez tout et - allez voir qui? « Notre distributeur » s'est transformé en « votre vendeur ». C'est n'est pas notre problème, c'est le vôtre, nous « souhaitons avoir répondu », « en salutations ».

Pour AMSTRAD l'affaire semble close et terminée. Nous, nous estimons que l'Association de Mr Rouichi devrait recevoir son coprocesseur et des plates excuses pour l'attitude cavalière du « service marketing ». Nous pensons que l'entreprise de la taille et de la réputation d'AMSTRAD devrait faire l'effort d'indemniser le client, induit accidentellement en erreur, surtout compte tenu du coût d'une telle réparation et du caractère Associatif du client.

De surcroît, ce qui nous intéresse c'est l'opinion des distributeurs qui doivent prendre en charge tout ce micmac d'AMSTRAD, faire face à des annulations de ventes et perte de clients mécontents. Et la concurrence, qu'en pense-t-elle? Le détournement de la clientèle, induite en erreur par le biais d'une publicité inexacte serait un intéressant sujet de débats...

Nous attendons les suites de cette affaire et réservons d'emblée nos pages au droit de réponse d'AMSTRAD.

❐

☞ ***« Passionné d'informatique depuis quinze ans, et lecteur depuis la première heure de la plupart des revues françaises consacrées à ce domaine, je suis consterné par la médiocrité desdites revues. Pour une meilleure compréhension du reste de cette lettre, je dois préciser que, professeur de lettres classiques par formation, enseignant dans le secondaire et le supérieur, je dispense actuellement la plus grande partie de mes cours en informatique. C'est la première fois que j'écris à une revue, et c'est à la vôtre, parce que DEBUG me semble un des seuls journaux qui ne se moquent pas de leurs lecteurs... même si je dois être encore très critique à son endroit. Je crois qu'il serait temps de poser clairement quelques questions fondamentales :***

Je suis hérissé par l'emploi continuel et abusif du mot « informatique » : l'informatique est - par définition - , la science de l'information, au sens linguistique du terme. Par conséquent, elle traite aussi bien de la grammaire, du codage génétique, du téléphone, de la cryptographie, du cinéma... bref, elle traite de tout. Lorsque je fais analyser à mes élèves le schéma transactionnel d'un document littéraire, je fais de l'informatique. Dés lors, l'importance de la machine est très secondaire, et l'intérêt de tel ou tel logiciel, quels qu'en soient les mérites, est purement anecdotique. Le rôle d'une revue d" « informatique » n'est-il pas aussi de dire cela ?

Pourquoi la plupart des journaux « techniques » sont-ils aussi mal écrits ? Je suis scandalisé par la quantité de fautes de Français, d'orthographe, de ponctuation, de lexique qui émaillent leurs colonnes. Il y a d'ailleurs contradiction : écrire un programme, c'est manifester une vigilance extrême dans le langage; comment peut-on y renoncer dès qu'il s'agit de commenter, de présenter, de réfléchir ? Bien sûr, il y a des coquilles, les erreurs, les oublis : nous y sommes tous sujets. Mais il y a aussi des fautes qui témoignent d'une ignorance crasse de notre langue... et vous êtes - tout de même - des journalistes ! Comment Mr. Hoffman, qui vient, il est vrai, d'une revue particulièrement gratinée en ce domaine, peut-il écrire une phrase comme

*celle-ci : « Cet octet est aussi important pour l'erreur qui est intervenue, la signification de chaque bit se divisant en erreurs du genre (...) » ? (N° 8, pp 24 sqq). Je suppose qu'in*tervenue *veut dire* survenue, *mais une* signification qui se divise en erreurs ???? *Je crois pouvoir penser qu'il voulait écrire :* cet octet nous informe aussi sur l'erreur qui s'est produite, chacun des bits indiquant respectivement (...). *(Heureusement , il nous rappelle page 28 le but pédagogique de ces articles !). Mais le bouquet, c'est que des articles traduits - par un traducteur , on peut l'espérer, et non par un amateur de passage - donnent la même sensation d'approximation ; je prends au hasard l'article de G. Korwin, dans le même numéro, et je lis :* cela résulte en des éditeurs américains contents... il n'y a qu'un seul constructeur... les rapports de force ne sont pas égaux... personne ne se pose la question de savoir... connaissez-vous des éditeurs qui ont attaqué... *et je ne parle même pas des mots inventés, des galimatias franglais omniprésents* (interopérabilité *pour* ouverture, librairie *pour* bibliothèque...), *d'une ponctuation totalement aléatoire...*

Alors, je me pose une question bien cruelle : quand une revue « pompe » le plus clair de son contenu dans une revue étrangère, cela semble indiquer que nous avons des journalistes bien légers sur le plan technique ; si de surcroît ils sont nuls en Français, on se demande ce qu'ils font dans ce métier ! Cela dit, il y a d'excellents articles, et c'est bien réconfortant.

Mais pourquoi trouve-t-on toujours les mêmes thèmes dans ces revues ? Grosso modo, si l'on supprime la publicité (merci de nous épargner !), tous ces journaux proposent des essais de matériels (sans aucun intérêt, vu que les appareils sont partout les mêmes), des essais de logiciels (en général inutiles, puisque, pour ménager l'éditeur du produit, on ne lui trouve que des qualités), quelques programmes (souvent largement inspirés - sinon plus - d'un ouvrage ou d'une revue de référence), et un peu de théorie, en général recopiée habilement d'un bon bouquin. Quand allons-nous trouver une revue pour mettre un peu d'air dans tout cela ? Pour expliquer à nos étudiants comment on pense un programme ?

Pour dénoncer l'escroquerie omniprésente de la recherche de vitesse (du style j'écris INC AX, INC AX plutôt que ADD AX, 2 pour gagner 3 millionièmes de seconde - alors que mon programme est tellement mal conçu qu'il perd 90 % de son temps à des opérations inutiles) ? *Pour montrer qu'un bon programme doit aussi être* beau ? *Pour démasquer la perversité de ceux qui considèrent qu'un programme doit être illisible, si l'on veut paraître brillant ? Pour déboulonner les arguments spécieux et pourtant partout répétés de vitesse, d'efficacité et de portabilité (!) du C ?*

Puis-je me permettre quelques suggestions de rubriques ? Il serait agréable de trouver une présentation bibliographique (éventuellement thématique) qui soit critique... c'est à dire qui annonce clairement qu'un bouqin est nul s'il l'est (il y en a beaucoup !) trop cher s'il l'est (c'est souvent le cas) et que tel autre est bien meilleur ? Je suggère aussi une sorte de « banque » de sous-programmes : chacun a sur sa machine quelques dizaines ou quelques centaines de procédures qui lui servent tout le temps, et qu'il a mises au point au cours des années... On peut imaginer qu'à chaque numéro, on lance un « tour de table » pour savoir quel lecteur a trouvé une solution élégante à tel ou tel problème. Une autre rubrique serait la bienvenue, surtout dans un journal comme le vôtre : un recensement - non exhaustif ! - des bugs les plus gênants dans les outils « standard », et la manière d'y faire face ; il m'est arrivé de chercher pendant plusieurs jours pourquoi telle variable avait un comportement curieux, avant de constater, en examinant le code produit, que c'était le compilateur qui déraillait dans un cas très particulier ; de même, il serait sage d'avertir le lecteur que certaines commandes de nos logiciels à tout faire sont de véritables traquenards.

J'espère que ces remarques, parfois acerbes, seront reçues non comme des attaques, mais comme des témoignages d'une volonté de collaborer pour améliorer encore votre revue. Croyez en tout cas à mes meilleurs sentiments... et continuez !

Bernard Jouët
Cholet.

☞ ***« J'ai suivi avec intérêt la polémique déclenchée par certains de vos lecteurs sur la qualité littéraire des textes publiés dans Debug Magazine.***

J'estime pour ma part que l'exactitude doit rester votre préoccupation essentielle, et que l'intérêt technique des articles proposés n'a pas à pâtir d'erreurs supplémentaires introduites par un éventuel correcteur, non spécialiste du jargon informatique ».
Jacques Vidal
Montpellier.

Nous publions ici la lettre de Mr Jouët car elle apporte, certainement, des remarques critiques d'intérêt général. Nous voulons apporter deux éléments de réponse suivants :

1. Mr Hoffman comme Mr Korwin sont tous les deux d'origine étrangère, américaine et transylvanienne, respectivement. Ces deux auteurs (comme par ailleurs d'autres auteurs étrangers qui publient dans DEBUG), ne « pompent » pas dans des revues étrangères, mais écrivent, tant bien que mal, aussi bien qu'ils peuvent, en français. Nous procédons à des rewriting approfondis de leurs textes, mais leur « couleur » particulière, qui donne cette impression de traduction, subsiste...

Comme nous avons déjà indiqué, DEBUG ne « pompe » pas dans des revues étrangères. Nous publions certains textes sous licence de Dr DOBB'S Journal, ce qui est la spécificité de notre revue. Ces articles reçoivent un bon accueil auprès nos lecteurs, comme peuvent témoigner d'ailleurs les résultats de notre sondage permanent - ils obtiennent toujours les meilleurs scores. Nous retenons avec intérêt vos propositions rédactionnelles. Une rubrique biographique est de plus grand intérêt : nous attendons vos textes... Nos pages sont aussi ouvertes à des « tours de table », rubrique « bugs »et la publications des bibliothèques des sous-programmes. A vos claviers!
la Rédaction

☞ ***PROTECT***

L'article PROTECT a suscité, visiblement, beaucoup d'émotions. Voici d'autres lettres que nous avons reçu à son sujet:

« J'AI DEVORE AVEC TOUJOURS AUTANT DE PASSION LE N° 10 DE DEBUG ET J'AI NOTAMMENT APPRECIE LA DIMINUTION DES FAUTES DE FRAPPE, DE MISE EN PAGE OU DE TRADUCTION. LA QUALITE TECHNIQUE ELLE EST TOUJOURS EXCELLENTE. BRAVO, CONTINUEZ !!

PAR AILLEURS, L'ARTICLE DE BRUNO VASSEUR SUR « PROTECT » A EVEILLE MON INTERET: POURQUOI DONC LE PROGRAMME PUBLIE DANS LE N°8 NE MARCHAIT-IL PAS ? JE VOUS PROPOSE EN PAGE SUIVANTE L'EXPLICATION (TOUTE BETE D'AILLEURS): LA SECRETAIRE CHARGEE DE TAPER LES LISTINGS COPIERA 100 FOIS » JE NE DOIS PAS CONFONDRE POP ET PUSH »...

Tout de suite à la lecture du listing un problème apparaissait: il y a plus de PUSH que de POP, ce qui n'est pas franchement recommandé. Il restait à localiser et expliquer le bug. Le mal se situe dans la procédure CHANGE_BASE. Dans un 1° temps on lit l'adresse de la DPT grâce à l'appel de l'interruption 1EH (qui n'en est d'ailleurs pas vraiment une). L'adresse se trouve alors en ES (segment) et BX (déplacement). A partir de cette adresse, il va falloir sauvegarder les 11 octets concernés à un autre endroit, puis les remplacer par les octets définis dans DISK_BASE. Cette manipulation va se faire avec l'instruction de chaîne MOVS qui prend une chaîne en DS:SI et la déplace en ES:DI.

C'est à ce moment que le problème va survenir: la chaîne à manipuler est actuellement pointée par ES:BX. Il faut donc charger ES en DS , BX en SI puis pour la destination ES avec CS et DI avec l'offset de SAVE. Tout cela est fait sauf pour DS qui n'est pas chargé avec la valeur contenue dans ES. Le listing correct serait:

```
PUSH ES
POP DS;là DS contient bien la valeur stockée dans ES
PUSH CS
POP  ES

au lieu de:

PUSH ES
PUSH DS
PUSH CS
POP  ES
```

En fait DS n'ayant pas été correctement chargée la zone sauvegardée n'est pas la bonne. Par contre dans l'étape suivante de réécriture de la DPT l'adresse de la table est la destination dans l'instruction MOVS et

doit se trouver dans ES (pour le segment). Or ES a été modifié depuis la lecture de l'adresse de la DPT, c'est en fait DS qui devrait contenir la valeur de segment de cette adresse d'où les instructions:

```
PUSH DS
POP  ES
```

qui doivent assurer le transfert. Seulement comme DS n'a pas été chargé avec la bonne valeur précédemment ES ne l'est pas plus et les 11 octets sont écrits ailleurs: la DPT n'est pas modifiée, le programme ne peut pas marcher et en plus quelque part en mémoire des octets qui ne demandaient rien à personne ont été modifiés sauvagement et risquent de se venger...»
J.CHALEARD
VILLEPREUX

« COMME CE N° DOIT PARAITRE FIN MARS, C'EST A DIRE LA VEILLE DU 1° AVRIL JE VOUS PROPOSE (S'IL RESTE UN PETIT PEU DE PLACE A COMBLER) UN TRES COURT PROGRAMME APPELLE « GAG » QUI FERA LA JOIE DE TOUTES LES SECRETAIRES.

« GAG » est un petit programme résident (une trentaine d'octets) que j' ai écrit pour que le 1° AVRIL de ma secrétaire ne soit pas trop morose. Je vous laisse découvrir ce qu'il fait. C'est assez amusant (à condition de ne pas laisser durer la plaisanterie trop longtemps quand même) et sans danger pour le travail effectué (pas de fichiers perdus ou abimés).

Pour l'utiliser, copiez le sur le disque de travail puis modifiez (après sauvegarde) l'AUTOEXEC.BAT de votre victime de manière à ce que « GAG » soit chargé dès le lancement de la machine. Demandez ensuite à votre secrétaire préférée de vous taper un courrier et attendez...

Avant qu'elle ne devienne tout à fait folle, replacez l'autoexec.bat d'origine et relancez la bécane pour remettre les choses dans l'ordre. Les instructions de compilation sont dans le listing ».
J.CHALEARD
VILLEPREUX

```
;*************************************************
                      G A G
;*************************************************
                        ;
;Programme résident qui détourne l'interruption 17h
;(envoie de car. à l'imprimante ) et transforme les
;caractères (ajoute 1 au code ASCII). Peut tester si
;déjà résident et se desinstaller (fonction
;interrupteur) en rendant valides les lignes
;commençant par ;++ et remplacer le jmp install en
;ligne 53 par jne install. ATTENTION: A ASSEMBLER EN
;.COM EXE2BIN PIEGE PIEGE.COM
;*************************************************;

code segment para 'CODE' ;Définition du segment de CODE
         org 100h
         assume cs:code, ds:code, es:code, ss:code

start:   jmp install


oldint  equ this dword ;ancien vecteur d'interruption 17h
ofsoldint dw (?);Adresse d'offset vecteur d'interruption 17h
segoldint dw (?);Adresse de segment vecteur d'interr. 17h


NewInt   proc far

         jmp  short NewInt_1


NewInt_1: or   ah,ah   ;Imprimer caractère (Fonction 0) ?
         jne  aint     ;NON -> Appeler ancienne interruption

         pushf
```

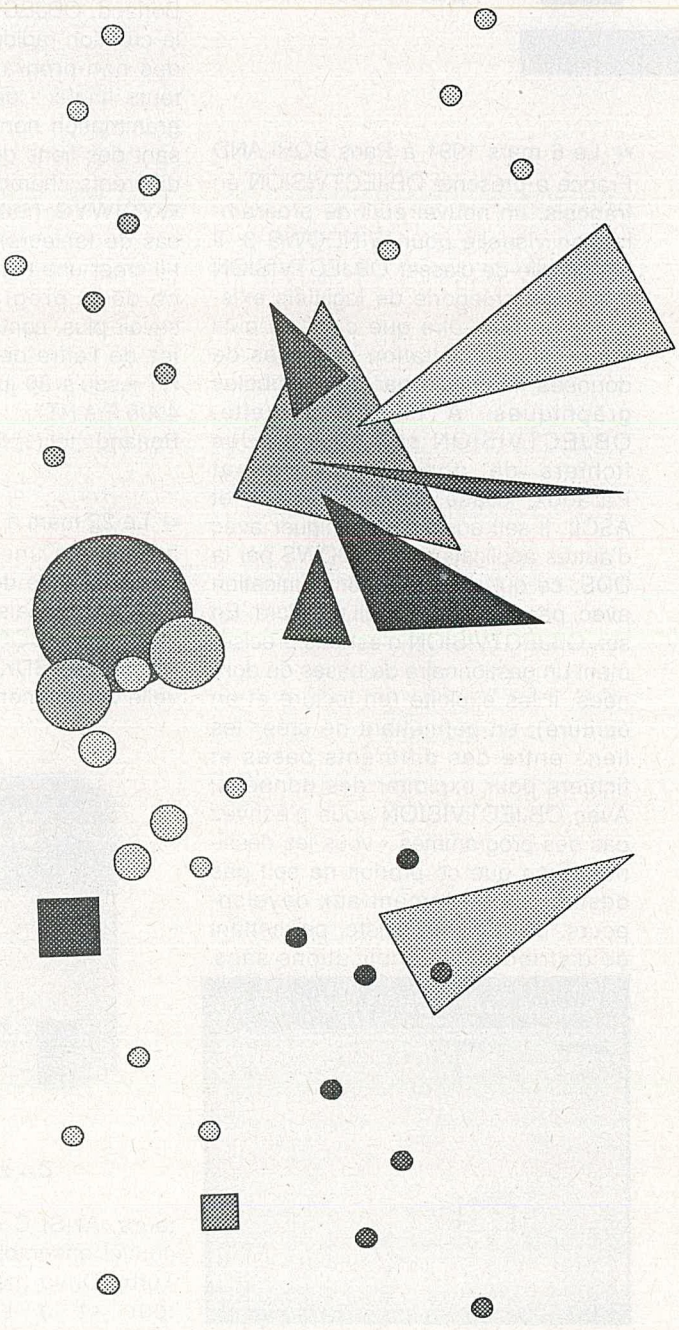
```
            cmp al,32
            jna  rien
            add al,1
rien:       popf

aint:       jmpcs:[oldint] ;Vers l'ancienne routine imprimante

NewInt      endp

Long_Pro    equ this byte


install     label near

   mov  ax,3517h ;Lire contenu du vecteur d'interruption 17h
   int  21h           ;Appeler fonction DOS
   mov  segoldint,es ;Ranger adresses de segment et d'offset
   mov  ofsoldint,bx ;Ranger vecteur d'interruption 17h


 mov  dx,offset NewInt ;Adresse d'offset nouvelle routine d'interr.
 mov  ax,2517h       ;Contenu du vecteur d'interruption 17h
 int  21h            ;sur routine utilisateur
 mov  dx,offset Long_Pro;Calculer nombre de paragraphes (unités de 16
 mov  cl,4           ;octets) dont doit disposer le programme
 shr  dx,cl
 inc  dx
 mov  ax,3100h       ;Terminer programme par code de fin 0 (o.k)
 int  21h            ;mais rester résident

;== Fin================================================
code        ends                    ;Fin du segment de CODE
            end  start
```

Joignez le dialogue!

Ecrivez nous à l' adresse suivante :

DEBUG MAGAZINE
3 rue de l'Atlas
75019 PARIS

Toutes les lettres deviennent la propriété de DEBUG MAGAZINE et ne peuvent pas être retournées. Nous nous réservons le droit d'abréger le texte pour des raisons d'espace (c'estsignalé par (...)).
La Rédaction



☞ *CONTACT USA*

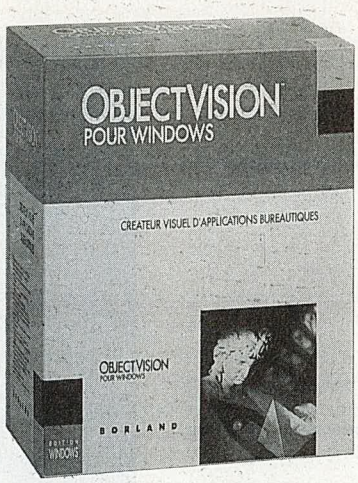
« Nous cherchons des développeurs, pour leur parler comment ils peuvent introduire leurs produits sur le marché américain. Nous recherchons les développeurs qui ont des produits innovatifs, de haute qualité. Nous pouvons aider ces développeurs à établir les liens rentables pour vendre leur produits en USA, par la voie de licence aux éditeurs américains ou par les contrats OEM, VAR ou de distribution. Nous savons qu'en France il y a des produits de haute qualité qui peuvent avoir du succès en USA.

Nous pouvons discuter aussi des aspects légaux américains: patentes, marques, protection par copyright, licences, contrats et autres problèmes administratifs et légaux qui surgissent quand on fait des affaires en USA.
« J' attends vos propositions »

Frederic D Abramson Esq.
USA Inc
United Software Associates
21155 Woodfield Road
Gaithersburg, MD 20882, USA
Téléphone : (301) 840 9733
Télécopie : (301) 869 5636

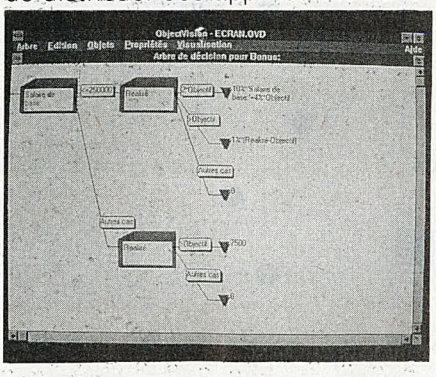
RUMEURS

par *ART*



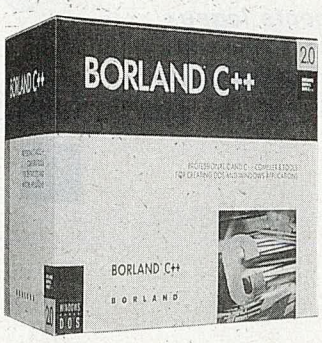
Objectivision de Borland

☞ Le 6 mars 1991 à Paris BORLAND France a présenté OBJECTVISION en français, un nouvel outil de programmation visuelle pour WINDOWS 3. Il est difficile de classer OBJECTVISION dans une catégorie de logiciels existants; on peut dire que c'est un outil interactif d'exploitation de bases de données, manipulé par des symboles graphiques. A l'heure actuelle, OBJECTVISION sait exploiter les fichiers de données au format Paradox, xBase, Btrieve (Novell) et ASCII. Il sait aussi communiquer avec d'autres applications WINDOWS par la DDE, ce qui assure la communication avec, par exemple, Excell ou Word. En soi, OBJECTVISION n'est pas précisément un gestionnaire de bases de données, il les exploite (en lecture et en écriture), en permettant de créer les liens entre des différents bases et fichiers pour exploiter des données. Avec OBJECTVISION vous n'écrivez pas des programmes - vous les dessinez. Bien que ce produit ne soit pas destiné expressément aux développeurs, un run-time existe, permettant de distribuer des applications sans l'outil de création des arbres de décision et formules de calcul. Selon Borland, OBJECTVISION va permettre la création rapide des applications par des non-programmeurs - les utilisateurs finaux - déjà habitués à la programmation non procédurale, en utilisant des liens de type déclaratif entre différents champs (objets) des écrans WYSIWYG (même démarche qu'en cas de tableurs). OBJECTVISION va-t-il créer une révolution dans le domaine de la programmation? Pour en savoir plus, contactez Borland et profitez de l'offre de lancement : 995 Frs HT jusqu'à 30 juin 1991 (prix normal : 4995 Frs HT).
Borland : tél (1) 39 46 96 69

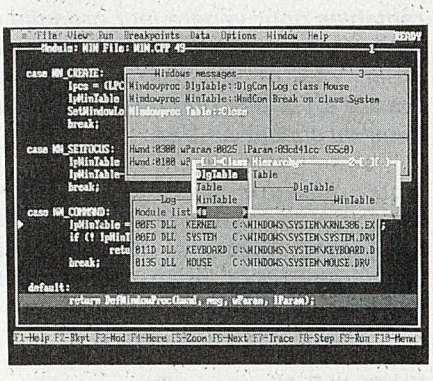


Objectivision : dessin d'arbre de décision

☞ Le 22 mars à Paris 1991 BORLAND a présenté une nouvelle version de son ensemble de développement C et C++ en français, permettant le développement DOS et WINDOWS sans recourir au SDK Microsoft. Cette nouvelle version comprend deux compilateurs, ANSI C et C++ AT&T 2.0. Le nouvel ensemble apporte notamment Turbo Drive (permettant au compilateur et à l'Environnement de Développement de fonctionner dans le mode protégé, dans une fenêtre WINDOWS, en mode caractère), le Whitewater Ressource Toolkit (un ensemble d'outils pour créer visuellement des icônes, bitmaps, menus et boites de dialogue), Turbo Debugger (avec l'unique option de debogage en marche arrière) et Turbo Assembler pour DOS et WINDOWS, les compilateurs des ressources, des entêtes et d'aide sous licence Microsoft et d'autres outils. L'environnement de Développement est, comme d'habitude, excellent. Il enchaîne automatiquement l'éditeur, compilateur, linkeur, debogueur et tous les autres outils nécessaires pour développer une application WINDOWS. Une fonction permet de forcer le code au strict respect de la norme ANSI, seules commandes conformes au standard Kerninghan & Ritchie ou la norme UNIX. Sous debogueur, vous pouvez voir votre code et son environnement sous plusieurs aspects, à travers de différentes fenêtres. Si vous disposez d'un processeur 386 ou 486, le debogueur permet d'utiliser les fonctions de debogage matériel du processeur. La fonction TDREMOTE permet d'exécuter l'application sur une machine et le debogueur sur une autre. Pour profiter de ce nouvel ensemble vous pouvez effectuer la mise à jour de votre Turbo C Professionnel et Turbo C++ Professionnel au prix de 995 Frs HT. La maj de Turbo C et Turbo C++ coûte 1490 Frs HT. Le prix public est de 4995 Frs HT.
Borland : tél (1) 39 46 96 69



C++ 2.Ø de Borland



Environnement Intégré de Développement c++2.0

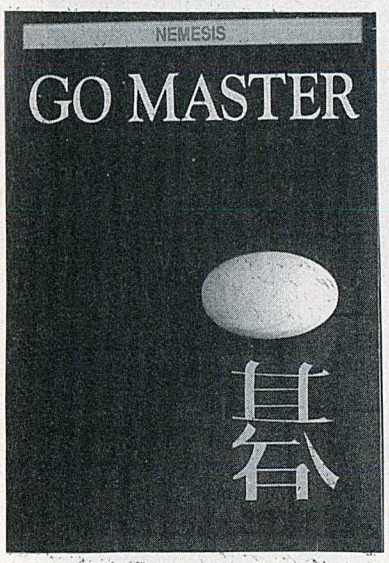
☞ Le 22 mars 1991 à New York BORLAND a présenté PASCAL pour WINDOWS. Cette nouvelle version, comme le C++ pour WINDOWS, élimine la nécessité d'utiliser SDK WINDOWS de Microsoft. De surcroît, elle simplifie à un extrême minimum le code nécessaire pour rendre l'application compatible WINDOWS. Pour qu'un simple programme d'affichage (de type writeln(' Hello');), il suffit d'ajouter une ligne

[SERVICE LECTEURS : INTERESSANT = 4 / MOYEN = 5 / MAUVAIS = 6]

d'appel au système, pour que le code compilé puisse être exécuté directement dans des fenêtres WINDOWS. Contactez Borland au (1) 39 46 96 69.

☞ Du 8 avril au 31 mai 1991 DIGITAL RESEARCH organise l'action « mise à jour vers DOS 5.0 » pour seulement 575 Frs HT (682 Frs TTC). Selon Digital Research, « DR DOS 5.0 est actuellement le seul système d'exploitation qui permette de tirer parti de toute la puissance des applications et des micro-ordinateurs modernes. Ceux qui décident une mise à jour vers DR DOS 5.0 obtiennent non seulement une mémoire conventionnelle de 620Ko complètement libre, mais surtout résolvent une très grande partie des nombreux problèmes de compatibilité résultant des multiples versions de MS DOS. DR DOS 5.0 est parfaitement compatible avec toutes les versions du MS DOS, Novell, Banyan, MS NET, PC LAN et 3COM ». L'ensemble de mise à jour (qui est totalement complet et opérationnel mais n'est pas auto-bootable) remplace un DOS existant par DR DOS 5.0. Pour bénéficier de cette offre, il vous suffit de contacter Digital Research au (1) 39 46 33 66.

☞ Réputé impossible à programmer, le GO est aujourd'hui disponible pour Mac et IBM PC chez Quartet Systems. Prix 780 Frs HT, tél : (1) 90 27 00 76.



Le jeu de GO sur Mac et PC

☞ MICROSOFT annonce ballPoint, une souris spécialement conçue pour les portables, se fixant au clavier, réunissant la technologie des souris et des trackballs. Elle coûte 1690 Frs HT, offerte gratuitement à tous les acheteurs de COMPAQ LTE ou COMPAQ SLT jusqu'à 30 juin 1991. Appelez Microsoft au (1) 69 86 46 46.

☞ BRIEF version 3.0 en français est annoncé par INNOSOFT. Il est personnalisable, en fonction de besoins de développeur, permettant de configurer le clavier et de créer des macrocommandes de type C ou LISP. Prix 3290 Frs HT. dBRIEF est un module d'adaptation au style de programmation xBASE (FoxBase, dBase, Nantucket, Quicksilver etc.). Prix 950 Frs HT, tel (1) 45 06 76 91.

☞ ProtoView Developement Co annonce version 3.1 de ProtoView Screen Manager Facility et version 1.1 de ProtoGen Application Generator. Les deux outils permettent de dessiner intéractivement fenêtres, menus et boites de dialogue et générer automatiquement le code C résultant. Prix : ProtoView $ 695.-, ProtoGen $ 199.-, ensemble $ 894.-; téléphone : 19,1 (908) 329 8588 fax (908) 329 8624.

☞ Rational Systems Inc annonce OXYGEN, un utilitaire qui permet à MICROSOFT C d'exploiter pleinement la mémoire étendue, sous DOS et WINDOWS. OXYGEN permet de compiler, linker et optimiser des programmes plus larges, sans partiels sur disque, sans quitter l'environnement de développement. Prix d'introduction $ 399.-, téléphone : 19,1 (508) 653 6006, fax (508) 655 2753.

☞ CANON, qui fabrique des millions des cartouches jetables pour les photocopieurs et imprimantes laser, va tenter de les récupérer pour le recyclage. Animé par un soudain réflexe écologique, CANON a construit une géante usine de retraitement de 88100 m en Chine Populaire, qui va recycler les cartouches récupérées dans le monde entier. Le recyclage n'est pas une simple réutilisation : les cartouches vont être démontées, les matières refondues et utilisées pour la fabrication des cartouches neuves.
Contactez CANON au (1) 48 65 42 23.

☞ Le 4 mars 1991 le juge William Ingram a déclaré que le sigle « 386 » est une désignation générique et ne peut être protégée par la loi sur les marques en USA. Cette décision ouvre à des concurrents d'INTEL, notamment à AMD, la voie libre pour commercialiser des clones 386. INTEL réagit, en changeant l'appellation de ses processeurs, qui désormais vont s'appeler « i386 » ou « Intel386 », pour distinguer les produits génériques Intel. Une autre affaire, portant sur le droit d'AMD à distribuer les produits incorporant le micro-code Intel, doit être jugée prochainement.

☞ L'Association « Généalogie Assistée par Ordinateur » annonce GAOPERSO - un programme de saisie et traitement généalogique ascendante et descendante et GAOACTES, un programme de saisie des actes en relevé systématique et création de tables filiatives. GAOPERSO est « limité » à 2 milliards d'individus, sans limite du nombre de générations. GAOACTES est adapté aux fiches de dépouillement « TRA », pour enregistrer des actes paroissiaux et d'Etat Civil. Les deux logiciels, selon l'Association, permettront de « sauvegarder le patrimoine national », ils sont déjà utilisés dans des nombreuses communes.
Prix : GAOPERSO 800 Frs, tél (1) 43 58 67 31.

☞ CONV convertit et échange des fichiers d'une vingtaine de traitements de texte PC, plupart formats MAC et d'une douzaine d'autres formats. Grâce aux appels directs aux lecteurs de disquettes, il est possible de lire sur un PC des formats « incompatibles » tels que Wang, Canon, Atari ou Apricot. D'autres formats sont accessibles par le câble série : Sagem, Exxon, Philips, Bull. Prix : 3000 Frs HT tél : (1) 4246 04 39

☞ ORDIGESTION annonce une mise à jour de OrdiPaye 3.60 qui répond aux nouvelles disposition de la Loi de Finances sur la Contribution Sociale Généralisée 1.1%.
Contactez Ordigestion au :
(1) 47 68 56 22

L TAGE CA HEE de L ES

Sur la scène d'une industrie dont les divas ont des performances irrégulières, Bill Gates est un personnage qui é

Entretien réalisé par
John Perry BARLOW
Traduit par
***Philippe DALLEAS** et*
Jean GUIBERT

[SERVICE LECTEURS : INTERESSANT = 7 / MOYEN = 8 / MAUVAIS = 9]

INTERVIEW

Alors que la rumeur selon laquelle Bill Gates serait un peu fauché, pour un milliardaire, est sans aucun fondement, Bill Gates se trouve être, à cause d'une marge bénéficiaire absurde, celui qui a le mieux réussi de tous les entrepreneurs dans le domaine de l'informatique. Au cours des quinze dernières années, depuis qu'il a été renvoyé de l'université de Harvard, et après s'être passionné pour le premier PC de MITS Altair, il a créé, avec un ami de jeunesse, Paul ALLEN, la société MICROSOFT, qui est devenue, par la suite, le plus grand éditeur connu de tout l'univers. Il est aux commandes d'un quasi-monopole des systèmes d'exploitation pour micro-ordinateurs et il produit une gamme d'applications hautement performantes et diversifiées. Tout comme l'industrie dans laquelle il s'est incrusté, il se ramifie comme du chiendent.

Contrairement à la presque totalité des ses collègues des premiers jours, Gates n'a pas été la victime de ce qui semble être la fatalité du rituel de développement des sociétés industrielles au-delà d'un certain seuil - c'est- à- dire, la relégation du père fondateur. De nombreuses compagnies finissent par ressembler plus à leur marché qu'à leur fondateur. Gates s'est débrouillé pour maîtriser le tempo de sa longue valse avec le plus extraordinaire des partenaires, la Compagnie IBM.

Et alors qu'il n'utilise plus de sparadrap pour maintenir ses épaisses montures d'écaille, il reste, au moins en tant qu'individu, que c'est un sacré bonhomme. Il se balance d'avant en arrière (1) quand il est impatient - ce qui lui arrive souvent. Il a un sens de l'humour dans le genre adoprol (2). Sur sa tête, une mèche folle de personnage de bande dessinée. Le plus riche « self made man » américain de moins de 40 ans n'en paraît pas plus de vingt; il est bien de sa personne, plutôt charmant.

Et pourtant, rien de mièvre dans son esprit. Bien qu'ayant eu l'occasion de fréquenter de brillants esprits, je maintiens que Bill Gates est l'homme le plus intelligent qu'il m'ait été permis de rencontrer.. tout au moins en terme de MIPS (3) à l'état brut. Ce CPU fonctionne à un rythme phénoménalement rapide. Rien qu'à le voir, on sent qu'il fonctionne.

Lors de mes premiers contacts avec Bill Gates, j'ai été agacé par la finalité à laquelle il consacre ses chevaux-vapeurs « neuroniques ». En public, il les utilise à la construction d'un « monde selon Bill Gates », sorte de cosmogonie faite d'auto-références complexes et en laquelle l'épingle du doute n'a pas de place.

D'une conférence de presse à l'autre, je l'ai vu affirmer, avec toute la candeur du monde, que toute interprétation de la réalité, susceptible d'être avantageuse pour Microsoft, ne pouvait être qu'exacte.

« OS/2 a été un immense succès » dit-il. L'alliance d'IBM et de Microsoft a constitué une forteresse inexpugnable. Les utilisateurs ne voyaient aucune différence importante entre Windows et l'interface Mac. Cela m'a pris quelque temps pour réaliser qu'en certaines de ces occasions, il était le seul dans l'assistance à ne pas s'apercevoir qu'il le descendait.

Et la pire des choses, c'est que ça prenait. Le réalisme officiel de Microsoft doit, à l'occasion, se référer à un monde Orwellien, mais c'est fait avec beaucoup de souplesse. Les difficultés d'OS/2 sont occultées, soudainement, par le succès de Windows, éclatant et, pourtant, surprenant. Quand WORD 4 pour Mac sortit, la quasi totalité de ceux qui avaient entre temps acheté Full Write ou Nisus, s'empressèrent d'envoyer leurs 75 $ pour la mise à jour. Pour une entreprise de cette taille, Microsoft semble étrangement humaine. Evidemment, Bill Gates sait surmonter ses désillusions.

Voir Bill Gates en action me laissait un sentiment mitigé quant à mon désir de l'interviewer. D'une part, il ne fait aucun doute qu'il est un personnage extraordinaire, l'être le plus puissant et, probablement, le plus détesté de la galaxie informatique. D'autre part, interroger quelqu'un n'appartenant pas à la « ligne du parti » me semblait mince. Ma direction était plus réservée que moi.

Néanmoins, je parvins au bout de ses doutes et obtins son assistance pour m'introduire auprès de Gates. Je réussis même à faire venir ma directrice lors de notre entrevue au Fairmont Hotel à San Jose, le jour de l'inauguration de la conférence Seybold.

Le matin, il dirigeait une table ronde avec John Warnock et Steve Jobs. C'était le nième épisode du feuilleton commencé un an avant, lorsque Gates avait annoncé son association avec Apple pour créer un standard de description de fontes et de pages, pour concurrencer PostScript.

Je n'étais guère motivé par ce à quoi je venais d'assister. Les questions posées à Gates par Jonathan Seybold et par l'audience avaient été noyées dans un fangeux magma d'auto-justification. Autant débattre avec le diable, me mis-je à penser.

Heureusement, la personne que je rencontrai s'est révélée être à la fois plus humaine et plus avenante. Il est certainement un coquin, et son sens de l'esthétisme et ses conceptions sont très éloignés des miens ; mais à la fin de l'entretien, j'en étais arrivé à respecter Bill Gates et, d'une certaine façon, à l'apprécier.

L'entretien commença par un désaccord :

Ce matin, Steve Jobs a fait une remarque intéressante en disant que NeXT était la BMW de l'informatique et que Microsoft en était la GM. Qu'en pensez- vous ?

L'analogie ne tient pas tout à fait, parce que, dans le cas de BMW et de GM, les deux fonctionnent au même carburant et roulent sur les mêmes routes. Par exemple, quelqu'un qui piloterait une machine 486 dernier cri avec beaucoup de mémoire et de disques, - disons, un Compaq - pourrait clamer que c'est une BMW, et ensuite, les Acer pourraient prétendre être la GM, tout simplement parce qu'ils sont sur la même route, avec les mêmes applications, les mêmes disques et le même carburant. En ce qui concerne Steve, il dit « Nous, nous allons créer un système routier tout neuf, un tout nouvel ensemble de fonctions, une toute nouvelle interface » : donc l'analogie n'est pas appropriée. Il prétend avoir vendu dix mille unités l'an passé, et DOS en était à environ quinze millions; c'est très disproportionné en volume ... Et ne croyez pas que je veuille dénigrer GM ou qui que ce soit...

Peut-être la DeLorean ?

Oui, la DeLorean était sur la route. Elle fonctionnait à l'essence, autant que je m'en souvienne ... Clive Sinclair avait une voiture électrique, c'est donc différent. Oui, l'idée de Steve était bonne. C'était une bonne démo. Et si je pensais que son truc ne pouvait être réali-

sé que sur sa machine, alors je me serais dit « En voilà un qui va se retrouver dans la situation de Mac! ». Mais la vraie question est la suivante : peut-on faire de ces choses, qui sont chouettes et avoir la compatibilité avec toutes les applications, la soi-disant idée-force qui est là-dedans ? Et ma proposition est « tu paries ? ». Pas de problème.

Autrement dit, vous pensez qu'en dernière analyse Windows peut procurer autant d'élégance d'exécution que les Mac et les NeXT ?

Je ne comprends pas pourquoi vous avez dit « en dernière analyse ».

Bon. Je vois que nous disputons des goûts et des couleurs. Mais quand j'utilise Windows, il ne me semble pas aussi souple. Je n'ai pas l'impression de trouver le même niveau d'intuition dans votre système que dans Mac, que j'utilise de façon habituelle. Et pour le peu d'expérience que j'ai de NeXT, je fais le même commentaire.

Au moins, utilisez-vous les applications?

Oui, j'utilise les vôtres, largement.

Alors il m'est difficile de comprendre pourquoi... Quand vous êtes dans Excel, comment pouvez-vous savoir sur quelle machine vous êtes ? Comment pouvez- vous le dire ? Gardez-vous un oeil rivé sur le logo en vous disant « Hmm! Pas aussi élégant! Pas aussi élégant! » ? Est-ce le logo qui vous gêne ?

Non. Mais chaque fois que je veux copier un fichier, quitter le programme, ou retourner au système d'exploitation... vous avez raison en ce qui concerne Excel ...

Vous parlez du shell. Il y a quelque chose que vous n'aimez pas dans notre shell.

Il ne me donne pas la même impression de souplesse.

Avez-vous souvent utilisé Windows 3.0 ?

Oui. Mais en tant qu'utilisateur de Mac, je n'affirmerai pas m'en servir tout le temps. Peut-être m'y habituerais-je. Mais je ne m'y sens pas aussi à l'aise.

Bien. Prenons le premier système graphique auquel vous ayez été confronté, ou la première Feuille de Calcul, ou le premier Editeur de Textes. Vous vous y êtes habitué. Si vous avez pris l'habitude du Mac, moi je vous dis « Continuez, Mac est une sacrée bécane. Apple va continuer à l'améliorer » Mais si vous avez commencé par utiliser Windows, et qu'ensuite vous passez à Mac, vous allez vous dire « Eh! où se trouve le système dynamique d'échange de données ? ».« Où sont toutes ces fonctionnalités que possède Windows et auxquelles vous vous êtes habitué dans cet environnement ? » Tout ce que je puis vous dire, c'est que c'est plus souvent le marché qui décide de ces questions, ça marche bien pour Windows, ça marche bien pour Mac, ce qui signifie, à l'évidence, qu'il y a des gens qui achètent les deux. Ce sont les deux principaux standards d'ordinateurs pour lesquels nous allons investir massivement en applications.

Pensez-vous que le marché privilégie la facilité d'utilisation ?

Mais que pensez-vous qu'il privilégie ? Il privilégie le meilleur produit ! Nous sommes dans un pays libre, les gens achètent ce qu'ils veulent...

Si le marché privilégie la facilité d'utilisation, il y a longtemps que, par exemple, Excel aurait largement dépassé 1-2-3, et que le Mac aurait depuis longtemps dépassé DOS.

Non, il n'est pas tout à fait exact qu'il y ait une tendance à sous-estimer la grande variété d'applications qu'apporte DOS. Cette variété est un avantage réel pour les utilisateurs. La facilité d'utilisation y participe - aller voir un collègue et échanger une disquette, par exemple. Sincèrement, pensez-vous que trimballer cette machine avec vous (le Mac portable), c'est une facilité d'utilisation ? Dans ce cas de figure, DOS est bien plus facile à utiliser.

Parmi les fabricants, il n'y en a qu'une trentaine à faire des machines pesant le tiers, et plus puissantes, et moins chères, et, ça aussi, c'est une facilité d'utilisation. Quant à 1-2-3, si neuf de vos collaborateurs s'en servent, la meilleure chose que vous auriez à faire, pour avoir de l'assistance et d'échanger des données, c'est d'utiliser 1-2-3. Oui, je pense que la facilité d'utilisation est un critère fondamental. Et c'est la raison pour laquelle Windows permet aux utilisateurs de faire fonctionner leurs applications sous DOS et d'échanger des données entre elles. La facilité d'utilisation est le critère ; nous devons être très prudents et ne pas la définir au travers de nos manies ou de notre seule expérience, parce que chacun d'entre nous a une tendance à se polariser sur un ensemble particulier d'éléments, liée à la façon dont il en a pris connaissance au départ. Voilà ma réponse : oui, Windows est une référence importante et nous allons continuer à la promouvoir. Je réagis négativement à votre comparaison, parce que, sur bien des points, Mac est meilleur maintenant, et que, sur bien d'autres points, c'est Windows. Tous deux ont un brillant avenir et ils devront faire dans le futur ce qu'ils ne font pas maintenant.

Qu'en est-il de NeXTstep ? Pour répondre à votre affirmation que le Mac est ce à quoi je suis habitué, je me sens une sérieuse envie d'aller m'acheter une NeXTstation simplement parce que j'en apprécie le système d'exploitation et son état d'avance dans la version 2.0. Il me semble très facile à utiliser, plus que le Mac, et il est très élégamment conçu. Croyez-vous à la possibilité que d'autres personnes auront la même réaction ?

Pas assez pour que je décide d'écrire des applications pour cette machine. Parier sur des plates-formes - c'est ce pour quoi je suis payé. J'ai parié sur Macintosh. Lotus ne l'a pas fait, pas plus qu'Ashton-Tate ou WordPerfect. Moi j'y ai cru. Steve est venu avec sa société. Nous avons continué à travailler sur nos applications Mac. Nous pensons qu'il s'agit d'une plate-forme très importante et nous allons continuer sur cette voie. Sur certaines autres plates-formes, dont une nouvelle apparaît chaque mois, nous avons décidé de ne rien faire, tout simplement parce que nous voulons faire mieux sur Windows, Mac et OS/2. Ce sont là nos trois plates-formes principales. Si je pensais pouvoir vendre quelque chose au niveau de Next, j'y affecterais des ressources. A ce jour, je n'ai pris aucune décision.

Etes-vous d'accord pour dire que votre développement sur Mac valait la peine, même si Mac n'avait qu'une toute petite part de marché - ce qui, je pense, est toujours le cas ne serait-ce que pour la maîtrise que cela vous a donnée dans le développement d'interfaces graphiques pour les utilisateurs.

En fait, c'est exact, mais c'est très rentable pour nous de vendre des applications Mac. Il y a une multitude d'avantages pour les utilisateurs dans la compatibilité obtenue en utilisant Word ou Excel sur ces deux plates-formes. Mais en tant que telle, c'était une excellente décision.

Il me semble que ce que permet NeXT et que Sun commence à offrir, c'est l'opportunité de développer en matière de réseau ... voire de groupe. Ceci alors même que Windows utilise un système Novell n'offre pas le même genre d'interactivité ...

A quoi pensez-vous ? Moi, je pense que ce ne sont que des mots ! Vous êtes-vous servi de Lotus Notes ?

Oui, vous avez raison. Lotus Notes est un bon programme.

Mais l'interactivité ? Les bits ne changent pas, en quelque sorte ? Quand Steve dit aux bits de changer, ils changent, et moi, quand je leur dis de changer, ils ne changent pas ? Soyons sérieux. Nous avons plus d'interactivité qu'ils n'en ont. C'est ainsi que sont conçus les réseaux. Prenez ce truc appelé NetBIOS, vous pouvez transférer des bits de n'importe quelle machine à n'importe quelle autre. Ça semble marcher pour les noeuds, pour InfoAlliance, pour le partage de fichiers, pour LANMan et pour Novell. Il n'y rien de particulier à cela - il n'y a pas d'élément fondamental qui permettrait aux gens d'avoir des applications coopérant sur ces plates-formes d'une manière qui serait impossible sur des systèmes centraux.

Ainsi vous pensez que vous pourrez développer du logiciel de groupe en tâche de fond sous Windows, sur un réseau Novell, aussi bien qu'on peut le faire sous UNIX.

Bien. Pour la relation client-serveur, il vous faut un système d'exploitation à usage général fonctionnant sur l'un de vos serveurs. De la sorte, vous pouvez garder vos serveurs Novell et partager vos fichiers Novell, mais Notes exige un serveur OS/2, de même que InfoAlliance, FileShare et SQL Server. C'est pourquoi il vous faut quelque part sur votre réseau un serveur OS/2. Mais les meilleures applications intergroupes sont écrites sur nos plates-formes tant pour des raisons techniques que pour l'opportunité de les mettre sur le marché.

John Markoff (du New-York Times) vous a posé une question lors de la conférence Stuart Alsop, que je souhaite reformuler parce que j'avais trouvé votre réponse intéressante : croyez-vous à une possible balkanisation des systèmes d'exploitation due au fait que des produits élaborés par diverses entreprises, qu'il s'agisse de NeXT ou de Santa Cruz Operation, ou de tout autre, viennent vous concurrencer sur ces mêmes plates-formes?

Non. L'avantage de pouvoir échanger des applications et des données est immense. Tant que nous pourrons procurer toutes les innovations répondant aux tendances de la mode du compatible et fondées sur le standard PC, nous collerons à ce qu'attendent les utilisateurs. Allez demander aux utilisateurs si la meilleure approche, en ce qui les concerne, est bien notre capacité à évoluer dans la bonne pâte. Allez demander aux concepteurs d'applications si disposer de cette plate-forme maîtresse de DOS n'a pas été pour eux la meilleure des choses, pour un unique développement et des ventes très nombreuses. Allez demander à LogOne s'ils sont prêts à quintupler leurs surfaces de vente pour faire le même chiffre d'affaires ? Il n'y a pas de risque de balkanisation - le standard porte en lui-même ses propres mérites. Les fabricants ne tenaient pas à voir un standard s'imposer. DEC, TI et Wang avaient leurs propres standards. Même IBM persiste dans l'idée que le leur devient le modèle par lequel la notion de compatibilité a été définie. Ça n'a pas aidé, loin de là, et ça a conduit à la situation suivante : nous, nous sommes parmi ceux qui s'en sont dégagés et avons cédé la licence DOS à de nombreuses personnes. Intel a rendu son chip disponible au plus grand nombre et le standard a été créé. Et c'est cela qui a permis à quelques nouvelles entreprises dans le domaine du logiciel, d'émerger, d'écrire des logiciels innovateurs et de considérer ces cinquante millions d'utilisateurs comme une clientèle potentielle. S'il y avait eu balkanisation, il n'y aurait pas de « grands » logiciels : ça ne tiendrait pas. La force du standard PC est aujourd'hui plus grande que jamais. Il n'y a que dans l'industrie que l'on puisse entendre « Oh ! Ça n'a pas l'air mal. Ça pourrait se vendre », « Oh! Que j'aime ce petit détail ! Ça va se vendre ! ». Vous pouvez dire la même chose à propos desTableurs ... Il y a des milliers de Tableurs avec des détails et des caractéristiques plaisants qui sont apparus sur le marché ... Mais qu'en reste-t-il ? Il y a Lotus et Borland qui ont de l'endurance, nous aussi ... Mais ça ne représente qu'une petite part du marché. Les trucs chouettes, ça ne suffit pas. Les gens veulent des standards, des références.

Ce matin, vous avez dit quelque chose d'étonnant au sujet du regard que l'on peut jeter sur le marché quand on y est parvenu en premier et quand on y établi un niveau élevé de présence. Pensez-vous que c'est pour cette raison que des « chouettes » Tableurs, commes vous dites, n'ont pas été capables de tailler une croupière à Lotus ?

Si vous arrivez le premier, toute la formation et l'éducation du marché vous incombent. Les gens le comprennent. Les fichiers, les logiciels additionnels, les livres ... les doigts des gens commencent à travailler d'une certaine façon - c'est une force d'inertie. Il faut quelque chose de très radical, un immense pas en avant, pour rompre ce couple d'inertie. Excel a réussi a s'approprier une part du marché, mais le problème majeur était que l'interface graphique n'était pas l'activité principale sous DOS. Sur le Mac, nous avons une part énorme du marché, tout simplement parce que l'interface graphique est le courant principal, et qu'en plus nous étions le premiere Tableur. Sous DOS, tant que l'IGU (4) était un créneau, Excel était un créneau, et même un sacré créneau, crédible en de nombreux endroits pointus. Maintenant que l'IGU constitue le courant principal, Excel a une opportunité pour exploser. En fait, tant que Lotus n'arrivera pas à sortir très rapidement un produit sous Windows, et que ce soit un grand produit, Excel ira se construisant une position très forte. Ils le savent et font de leur mieux. C'est de la saine concurrence. Etre là le premier - et si l'on fait du bon travail - cela bloque toute influence sauf celle des améliorations des produits originaux.

Ceci me fait penser à Adobe. Ce que j'ai retenu des dires de John Warnock, en l'espèce, c'est qu'il proposait PostScript en tant que nouvelle norme ASCII. S'il y parvenait, ne deviendrait-il pas plus simple, pour les utilisateurs, de posséder plusieurs systèmes d'exploitation, étant entendu que des données extrêmement denses deviendraient parfaitement portables d'une plate- forme à une autre ?

Bon. Un système d'exploitation fait fonctionner des applications. Tant qu'il n'y a pas de standard de gestion de mémoire, de multi-tâche, de câblage, de systèmes de fichiers, de protection, et tous autres détails, on ne pourra guère vulgariser les systèmes d'exploitation, tout simplement parce que les systèmes d'exploitation ne sont pas figés. John lui-même reconnaît que le standard le plus important n'est pas celui dont il parle, les standards de données brutes (5), mais plutôt les standards de description d'objets (6). Ceux-ci sont devenus ce qu'il y a de plus important. C'est pourquoi il le fait là où il y a l'encodage idoine. Ce qui a maintenant le plus d'importance n'est pas tant de voir à l'intérieur des données que de s'accorder sur les renseignements les concernant. C'est là une innovation que nous introduisons dans nos systèmes d'exploitation. C'est du « top niveau ». Beaucoup y arrivent, certains mieux que d'autres, mais il n'est pas sain de dire « Ok. Nous allons tous le faire de la même façon ». Compte tenu de la position de nos systèmes d'exploitation, je pense qu'ils ont la meilleure chance d'avoir un réel impact.

A cause de leur part de marché établie et institutionnalisée ?

Et parce que je pense que ce que nous faisons est superbe. Parfaite synergie.

Avant Windows 3, quand l'avenir semblait appartenir à OS/2, un développeur m'a dit lui prédire un énorme marché, parce que c'était un produit Microsoft ; mais que, lui, il n'allait pas se mettre à développer des applications pour un système d'exploitation dont les manuels pesaient plus que lui. Qu'en est-il de la facilité de développement ? Si d'autres systèmes sont plus propices au développement, cela ne mettra-t-il pas Windows dans une situation désavantageuse ?

Dieu merci, ce n'est pas vrai.

Trouvez-vous Windows facile pour le développeur ?

Je ne crois pas qu'il existe de systèmes d'exploitation rendant le développement facile. C'est facile de développer des applications faciles et, dans le cadre d'une entreprise, c'est merveilleux. Vous pouvez prendre des outils comme ToolBook, Excel, Choreographer ou Easel - il y en a des tas - tous disponibles sous Windows, pour écrire des applications faciles. Ce sont de bons trucs pour les entreprises. Mais si vous voulez être concurrentiels et écrire le meilleur logiciel de PAO ou deTraitement de Textes, restons sérieux. Je me fiche de savoir de quel système d'exploitation il s'agit, car il faudra aller au charbon, écrire du code en dur, et tester, et rendre rapide et puissant. C'est du sport, et personne ne peut prétendre que c'est facile d'écrire un logiciel à succès dans son environnement. Nous faisons des tas choses pour rendre aisé le développement sous Windows, ceci en termes de meilleurs outils venus et à venir. Il y en a qui sentent l'opportunité et se disent « Tiens, il pourrait y avoir de meilleurs outils pour Windows », et ils les créent. Et il les vendent. Borland va venir avec ses produits, et nous aussi, et des milliers de jeunes entreprises font et ont fait un super boulot.

Vous me disiez, à Laguna Niguel, que vous ne considériez pas Windows comme un repli tactique par rapport à Presentation Manager; et pourtant, il ne semble pas que vous teniez le même langage qu'il y a un an ou deux, alors que vous décriviez un futur basé sur OS/2. Vous semblez maintenant parler d'un futur basé sur Windows.

Bon, Windows va fonctionner sous OS/2, et ainsi OS/2 est le futur. Ça n'a pas changé. Faire tourner les API (7) de Windows par dessus OS/2, ça a pris beaucoup d'importance pour nous, rendant aisé d'aller de Windows vers le noyau le plus sophistiqué d'OS/2. Mais nous avons fait totalement passer le message Windows depuis 1983. IBM ne faisait pas passer le message Windows parce qu'il n'en a pas pris la licence, et, de ce fait, bien des gens dirent « Ok. Microsoft considère Windows et OS/2 d'égales importances. IBM ne considère qu'OS/2 ». Et voilà, ceux qui pensent qu'IBM a la primauté dans la définition des standards, se disent probablement « Oh, on marche avec OS/2». Aux gens qui ont observé ce que nous faisions - et ceci comprend Lotus, WordPerfect, Software Publishing - nous disons « Voilà, toutes nos applications tournent maintenant sous Windows ; nous avons toujours été clairs là-dessus ; nous pensons que c'est la meilleure stratégie, que c'est le point de départ ». Certains l'ont fait, d'autres non. Et vu le succès de Windows, ceux qui nous ont suivis ont fait de belles affaires. Les autres y viennent.

Personne ne s'en tient à l'écart. A ma connaissance, il n'y a pas de développeur important, excepté peut-être Claris, qui ne développe d'applications pour Windows.

Par simple curiosité, pourquoi avez-vous mis tant de temps à porter Word sous Windows ?

Ça représente beaucoup de travail et on a envie de sortir quelque chose de nouveau. L'idée d'intégrer des tables à un Traitement de Textes, je l'ai eue pour Word sous Windows, il y a cinq ans. Le projet date de cette époque. Nous avons inventé la règle et le ruban il y a quatre ans, l'idée d'empiler les fichiers les plus récemment utilisés sur ce « fichier-corbeille »", l'idée d'insérer des ordres, c'était il y a quatre ans - au moment où nous avons enfin lancé l'affaire, j'avais oublié toutes ces chouettes astuces que nous avions créées. Puis j'y suis revenu et je les ai utilisées davantage en me disant « Tiens, c'est impressionnant comme toutes ces idées que nous avons eues ont été innovatrices ! ». En fait, les tables ont d'abord fait apparition dans les produits pour Mac et elles ont été bien accueillies ; maintenant, tout le monde veut des tables ... Mais c'était une première, et nous avons bien joué notre coup. Cela prend beaucoup de temps pour atteindre la bonne performance et la bonne adéquation, et tout en s'assurant qu'on ne dévie pas... Nous avons un magnifique Traitement de Textes en mode caractère - alors pourquoi faire Windows Word si ce n'est pas un grand pas en avant, par rapport à ce qu'on faisait dans le domaine des caractères ?

Ce matin, nous avons parlé de ce que vous avez fait personnellement. Je me disais que parmi les grandes entreprises, vous êtes avec John Warnock les derniers ingénieurs PDG. Avez-vous toujours la possibilité d'exercer vos talents?

Tout à fait. En matière de définition de produits, d'algorithmes... C'est même ce que je préfère à toute autre chose. Vous devriez rajouter Philippe Kahn à cette liste. Qui a commencé le premier, Adobe ou Borland ? Borland, j'en suis presque certain.

Borland a commencé dans un garage bien plus petit, c'est pourquoi il a mis du temps à se faire connaître...

Exact, c'est même le plus fort de nous

tous pour poétiser ses débuts.

Oui, ses débuts sont plus faciles à poétiser...

Non. Etre viré de l'université, quand il n'y pas de travail, pour aller chez MITS? (8). Ils n'avaient pas d'argent, ils étaient en faillite, on n'était pas payé ... Nous dormions à la belle étoile. Nous nous sommes développés à la campagne. Qui a créé le « Homebrew Computer Club » (9)? Nous sommes allés dans la « charrette »de MITS, y avons récupérés plein de types, avions fait tout ce qu'il fallait pour lancer des groupes d'utilisateurs. La puissance qui résidait dans cette charrette MITS pouvait faire tout péter, et nous vola la bande papier du BASIC, tous le monde nous volait notre BASIC, on n'arrivait même pas à se payer l'avion pour rentrer chez nous ... J'ai écrit une lettre au *Computer Hobbyist* en ces termes : « Hé! il ne faut pas pirater le logiciel ». Et ceci à une époque où Philippe étudiait encore les mathématiques en France, que Steve essayait d'être un baba-cool en Inde, et que Warnock était planqué dans un laboratoire chez Xerox. Nous avons tous une belle histoire à raconter.

Oui, il est totalement faux de dire que vous avez débuté avec IBM.

Pas du tout. Quand IBM est venu me voir, j'avais une part de marché bien supérieure à celle de maintenant. Je crois que je m'en serais bien mieux tiré s'ils n'étaient jamais venus me voir. Ce fut un désastre ! Que c'est-il passé? [éclat de rire] Il y a quelque chose d'étrange. Quelles étaient les principales machines de 1975 à 1980 ? Apple][avec notre BASIC, Radio Shack avec notre BASIC, Commodore avec notre BASIC. CP/M avait alors une petite part de marché ; nous y avions une part substantielle, mais c'est nous qui adaptions CP/M aux machines des uns et des autres. Digital Research n'avait pas envie de se salir les mains, alors c'est nous qui avons fait le boulot. Quelle était la forme de CP/M la plus répandue à l'époque ? C'était un produit que nous avons appelé SoftCard (10). Nous étions le plus important licencié de Gary Kildall avec une carte Z80 qui se branchait sur Apple][... C'est là le rôle que nous avons joué à cette époque ... il n'y avait pas grand monde. Nous avions une position bien plus forte. Si IBM nous a contactés, c'est qu'il y avait urgence. Ils devaient apporter la preuve qu'ils étaient capables de sortir quelque chose en un an, une machine pour laquelle ils avaient un objectif de vente de 20.000 unités. Ils voulaient sortir une machine 8 bits, y intégrer notre BASIC, et prouver qu'ils pouvaient au moins faire ça. C'est de notre faute si cela devint une machine 16 bits ; et nous avons été contraints d'écrire toute une nouvelle génération de logiciels. Et nous avons su utiliser tout ce travail comme base de définition d'un standard rigoureux.

Comment cela s'est-il passé ?

Celui qui travaillait sur ce projet était ouvert à ce changement. Les gens d'IBM passaient leur temps à nous demander de faire des comparaisons avec Appel][, nous passions le nôtre à leur répéter « Si c'est du 8 bits, ce n'est pas une nouveauté ». Intel positionna le 8088 à un prix très bas, limite. Il y avait quelques types sur le projet - ils étaient treize - qui voulaient intégrer une disquette. Dès qu'elle fut mise là-dedans, nous allions être riches. Parce que dès qu'il y avait disquette, nous pouvions leur prouver qu'il leur fallait un DOS, qu'il leur fallait plus qu'un BASIC, et que, certes, BASIC pouvait être en ROM, mais que DOS était la clé de voûte... C'est une longue histoire.

Quand vous avez commencé à faire des affaires avec IBM, aviez-vous l'intention d'écrire votre propre DOS?

Non. Ils sont venus avec une machine 8 bits avec un graphisme pourri, et voulaient BASIC. Nous leur avons répondu « N'utilisez pas le 8085, prenez le 8088 ». Et j'ajoutai « Oh, il y a cette société qui s'appelle Digital Research ». Il n'en avaient jamais entendu parler. J'ai téléphoné à Gary pour lui dire qu'ils allaient venir. Ils y allèrent, mais Gary étant absent, son épouse n'a pas voulu signer les protocoles d'exclusivité...

Alors, vous avez éconduit IBM ?

Oh que non ! Nous leur avons dit « D'accord ! nous allons écrire le langage et tout le reste .... Mais attendez un peu ». Gary n'avait pas de version 16 bits. Le seul système d'exploitation 16 bits disponible à cette époque était ce truc de Tim Patterson, qu'il avait appelé QDOS, « Quick and Dirty OS » (11). Comme Gary n'était pas intéressé, et que nous pensions qu'il se trompait, vu le court délai imparti et la position stratégique du système d'exploitation, nous décidâmes de le produire. Nous avons dit à IBM « Ne vous mettez pas martel en tête, nous nous chargeons du travail ». Nous nous entendîmes sur 200000 à 230000 lignes de code, ensuite on a acheté QDOS et débauché Tim Patterson de chez Seattle Computer Products. Tim nous a rejoint, et avec trois de nos collaborateurs, il fit DOS version 1. Puis, en cours de partie, environ 4 mois avant le lancement de la machine, Gary dit « Eh! Messieurs les IBM, il faut me mettre dans le coup ». « D'accord », répondit IBM, « nous lancerons aussi CP/M-86 ». Il leur fit alors un ensemble avec CP/M-86 et fixa un prix très élevé. Nous circonvînmes les ISV (12) pour les amener à nous préférer. Ça a pris deux ans et demi, avec l'appui de Wang, Zenith et Victor (oublié depuis lors), pour nous ouvrir une brèche. Et une fois qu'il y a une brèche ...

Revirement positif !

Je pense que nous avions une balkanisation. Une magnifique balkanisation. Nous avions trois systèmes d'exploitation, nous avions DEC avec un meilleur clavier, TI avec un meilleur graphisme et toute cette balkanisation. Et comment réagit le marché à cette balkanisation ? Il lui donna le baiser d'adieu.

Vous êtes dans le vrai. Les gens veulent du standard. Ils veulent des habitudes.

Et est-ce que ça va revenir ? Je ne vois pas pourquoi ça reviendrait. Je veux dire, posez-vous la question au fond, y a-t-il quoi que ce soit que vous ayez vu sur ces machines non-PC qui ne serait beaucoup plus beau sur un PC ? Et laissez à l'écart votre sentimentalisme.

Il y a de mystérieux aspects d'exécution, que j'ai beaucoup de mal à définir, mais que néanmoins je sens présents. Entre le Zen et l'Art d'Entretenir une Moto - vous ne pouvez pas définir la qualité, mais elle s'impose quand vous y êtes confronté.

C'est une perception plutôt intuitive.

Oui, je dois reconnaître que c'est un grand avantage. Bien des gens ne connaissent rien en technique. Leurs décisions d'investissement sont fondées sur l'aspect familier du produit. Si vous jouissez d'une

grande réputation, alors les gens achètent. Je voudrais insister sur un point - il est sans doute quelque peu mythique que, quoi que vous ayez acquis auprès de Seattle Computer Products, une grande partie de DOS était déjà faite.

Tim a été à l'origine de DOS version 1. C'est lui qui a écrit la majeure partie du DOS 1 que nous avons livré. Mais c'est en tant que collaborateur de Microsoft qu'il a réalisé le plus gros de ce travail. Le matériau que nous reçûmes avec lui, QDOS, possédait de sérieuses insuffisances . Mais quand nous l'avons livré, c'était un excellent produit. Il nous a fallu écrire une montagne de nouveau code pour réaliser MS- DOS version 2. Tim a pris part à cette version, puis il a réalisé un compilateur ; dans la version 3, il n'était pas impliqué ; la version 4 a été préparée chez IBM, et pour 5, Tim était détaché sur un autre projet.

Ça me fait penser à votre version d'OS/2 et à celle d'IBM. Pensez-vous qu'éventuellement OS/2 2 et OS/2 3 se développeront avec des concepts complètement différents ?

Non. Absolument pas.

Sont-ils condamnés à être similaires jusqu'à, au moins, la cinquième ou la sixième édition ?

Quand 3.0 apparaît, il remplace 2.0. Il remplace 2.x. Jetez tout ça. A un instant donné, il n'y a qu'une version qui compte. Avant 3.0, jusqu'à la sortie de 2.0, c'est 1.x qui compte. Quant 2.x est lancée, c'est 2.x qui compte. Quand 3.x sort, c'est elle qui compte.

Ainsi, vous travaillez à la version 3, tandis qu'IBM en est à la 2.

C'est une affirmation trop tranchée. Il y a des gens d'IBM travaillant chez Microsoft à la version 3. Il y a des gens d'IBM travaillant chez IBM à la version 3. La plupart de ces gens sont sur le site primaire, c'est-à-dire Microsoft, et sont des salariés de Microsoft. J'ai environ quarante personnes de Microsoft travaillant sur la version 2 ; ils passent beaucoup de temps chez IBM, les aidant pour la version 2. Il y a une grande coordination pour tout ce qui se passe à propos de la version 2.0. Alors qu'à l'origine nous étions chacun à parts égales, maintenant nous avons créé le concept de site primaire. De ce fait, chaque compagnie est non seulement licenciée pour tout ce qui est fait, mais, en plus, elle est très impliquée dans tout ce qui est en cours.

Mitch Kapor a dit que nous étions bien engagés dans l'ère de l'ordinateur, mais que nous n'avions toujours pas notre Henry Ford. Bien que ceci soit exact, il est bien possible que nous ayons notre John D. Rockefeller et la Standard Oil en votre personne et en Microsoft. Que pensez-vous de l'éventualité de voir Microsoft devenir la Standard Oil de l'informatique du 21ème siècle ?

C'est une question assez vicieuse.

Ce n'était pas dans mon intention. Je ne pensais pas à la Standard Oil en tant qu'hydre ...

A quoi d'autre pensiez-vous alors ?

Eh bien, à quelque chose de très, très efficace...

Monopole. Qu'est-ce qui faisait l'efficacité de Standard Oil ?

Ça marchait extrêmement bien. A telle enseigne que le gouvernement l'a démantelée.

Un monopole qui contrôlait les transports. Je n'y vois aucune analogie. Tous les secteurs où nous sommes sont extrêmement concurrentiels. Le Traitement de Textes. Les réseaux. Si j'en cite un qui ne soit pas concurrentiel, dites-le moi. Les Tableurs.

Les systèmes d'exploitation.

C'est un marché très concurrentiel. Il y a énormément d'innovations. Digital Research, voyez leur pub, voyez ce qu'ils font. Voyez ce que font ces centaines de types avec UNIX, qu'ils soient chez Mach, Open Software Foundation ou Santa Cruz Operation.

Mais, en termes de parts de marché, ils ne sont pas concurrentiels. J'ai l'impression que vous percevez principalement la compétition en parts de marché.

Pas du tout. Supposons que j'augmente le prix de DOS. Ce faisant, je perdrais très vite ma part de marché. Nous devons avoir une politique de prix très agressive et offrir continuellement des améliorations. C'est la raison pour laquelle nous affectons des ressources dans la nouvelle version de MS-DOS aussi bien qu'aux nouvelles versions de Windows. J'ai toujours eu conscience que c'est dans les systèmes d'exploitation que nous avions notre plus forte part de marché. S'il devait y avoir une analogie avec la Ford T, eh bien il s'agirait des machines MS-DOS, car il y a des centaines de fabricants de machines compatibles. Ce qui signifie, en termes de liberté de choix, qu'on peut choisir son matériel et utiliser toujours le même logiciel. Pour moi, cela constitue une contribution fondamentale. Ça a créé un standard qui a entraîné un très vaste marché. Si Ford a bien contribué à l'industrie automobile, c'est qu'il a décidé de faire un produit créant un très vaste marché, d'amorcer la pompe, de rendre le produit attractif. Je le pense quant à MS-DOS - c'est notre produit, mais il faut remercier ici IBM, Compaq, Zenith et toutes les autres compagnies et développeurs de logiciels qui se sont attelés à MS-DOS. Ils en ont fait un standard, c'est ce qui a fait la différence. Nous ne serions pas allés bien loin sans cette base.

Si IBM n'avait décidé de construire le PC, est-ce que DOS serait devenu un standard ?

Bien sûr que oui ! Nous aurions pris une autre machine comme modèle de compatibilité. C'est évident. En 1975, nous avions écrit que nous voulions un standard binaire fondé sur un jeu d'instructions. Le niveau de portabilité que nous avions dans les années 75- 80, c'était les programmes écrits en BASIC Microsoft, ainsi y eut-il des gens qui produisirent des logiciels de comptabilité et des livres de jeux. Quand on a appris notre BASIC, on le connaît. C'est un certain niveau de communication. Mais, dès le premier jour, nous savions que c'était inadapté. Dans le domaine des 8 bits, non seulement nous souffrions d'insuffisance de mémoire, mais encore avec les 6502, Z80, 6800 nous avions cette multiplicité de microprocesseurs. Le 8088 a été l'occasion de changer cela. En fait le PC d'IBM n'a pas été la première machine à base de 8088, il y avait une machine que nous avions faite avant celle-là, à partir de Zenith et de Mitsubishi. Nous nous sommes battus pour ce standard. Ce n'est pas IBM qui l'a fait, vous le savez - ils ont testé des tas de gens qui, selon eux, avaient emprunté des raccourcis pour fabriquer des BIOS et autres choses de ce genre. C'est utile d'être épaulé par une puissante société pour modeler un standard. Et aurions-nous dû prendre DEC, ou HP ou qui sait qui ? Et là,

nous entrons dans le domaine des conjectures.

Dans les années 80-81, j'avais décidé d'acheter un ordinateur. J'ai regardé ce qu'il y avait de disponible, et j'ai décidé d'attendre qu'IBM en sorte un, tout simplement parce que je voulais un produit élaboré par une firme sûre. Il m'a paru que pas mal de gens avaient agi de la sorte. Ils ne voulaient pas d'Apple. Ils ne voulaient pas de Victor.

IBM tient environ 30% du marché DOS. Donc, certains ont fait comme vous, d'autres non. IBM a joué un rôle très important et fait un excellent travail.

Je reviens aux systèmes d'exploitation. Je savais qu'il existait de nombreux systèmes d'exploitation, mais je ne savais pas les comparer. Je pensais qu'IBM allait définir un standard.

Non. Tant qu'on n'était pas débarrassé de la question du standard matériel, on ne pouvait pas avoir la compatibilité binaire. Le système d'exploitation en est un élément. Croyez-moi, quand ils ont introduit le PC, il y avait trois systèmes d'exploitation. Ça les amusait. Lequel préférer ? Lequel promouvoir ? Ils étaient dans l'embarras.

Quel était le troisième ?

L'UCSDP SYSTEM. Il était alors le seul système qu'ils utilisaient sur leur DisplayWriter. Ainsi y avait-il des systèmes d'exploitation qui s'entre-croisaient.

Vous sentez-vous investi d'une mission?

Oui.

Laquelle ?

Le PC est l'outil de l'ère de la communication. *Un PC sur chaque bureau et dans chaque maison.* Utilisant de superbes logiciels. Et les standards de logiciel pour y parvenir.

Qu'est-ce qu'il y a de bon à ce que cela advienne ?

C'est une question d'ordre théologique, et il y a des gens qui pensent que ce serait mauvais. Il y a différentes manières de considérer cela. Il se trouve que j'aime travailler à cette mission. J'ai réussi à m'entourer de collaborateurs exceptionnels, ça a été fantastique. C'est ce que je fais. Libre aux autres de dire si cela leur convient ou non.

On dit beaucoup d'âneries à propos de l'informatique personnelle. Beaucoup en parlent par rapport à son influence sur la société et sur l'individu. Avez-vous une opinion précise là-dessus ?

Moi, je pense que c'est très positif. Je vois ce qui ce passe dans ma société, comment les idées viennent aux gens. En ayant accès aux informations, ils peuvent envoyer des messages, à moi comme aux autres, provoquer les choses. Tout ce qui ressortissait à la spécialisation - penser aux manuels, aux contrôles - est maintenant à portée de main. Non seulement vous pouvez travailler vite, mais vous avez des gens, vous en tirez une plus grande créativité. Ils savent ce qu'ils font et la place qu'ils occupent dans l'ensemble. C'est stimulant d'être impliqué dans l'élaboration d'outils qui, à mon avis, rendent le travail plus intéressant. Je crois que cet âge de la communication est une belle chose. Je pense que cela rend les gens plus forts, le monde plus riche. A chacun de faire son choix. Moi, je pense que c'est bien. Et pas seulement parce que j'aime le faire - selon ma religion, c'est bon.

A propos de structures, je constate que votre entreprise a une structure très horizontale, fondée sur la melec (13). Apple a le même type d'organisation. Pourtant, chez vous ça fonctionne, mais pas chez Apple. A quoi est-ce dû ?

Je ne suis pas un spécialiste en ce qui concerne Apple. Apple a beaucoup de succès, mais c'est une entreprise différente de Microsoft. La rotation de leurs dirigeants est importante, et quelle qu'en soit la raison, je sais que l'incroyable stabilité de notre équipe dirigeante nous a été très précieuse. Nous allons en séminaire plusieurs fois par an, et on discute : « Eh! Où en sommes-nous dans la messagerie ? Faut-il nous lancer dans le conseil ? Que faire avec le canal de distribution? Notre approche du développement est-elle parfaite ? Où en est notre image - gros, c'est mauvais - que pouvons-nous faire pour montrer les belles choses que nous faisons ? Comment travailler avec les développeurs? ». On sort, on réfléchit à tout cela, et ça nous plaît ... Nous ne parlons pas de nos succès, mais de ce que nous aurions pu mieux faire. Voilà la culture Microsoft.

Vue de loin, Microsoft fait penser à une tribu et Apple à une machine, en ce sens que Microsoft possède un dirigeant mythique, en votre personne, ce qui n'est pas le cas pour Apple.

Oh, ils ont John Sculley, qui est, je crois, un très bon patron. C'est dommage de ne pas avoir vu ce que Steve Jobs apportait à cette société. C'était de grande qualité. Apple est une société très importante. Ils ont massivement investi en recherche et développement durant les quatre dernières années, et leur avenir dépend de leur aptitude à concrétiser. Je crois qu'ils disposent d'une sacrément bonne matière qu'ils vont faire connaître, et je leur souhaite plein succès parce que je pense que c'est d'une dynamique très positive que d'avoir un autre standard, parvenu à masse critique, poussant en avant et prenant des risques. Certes, nous avons conseillé à Apple de prendre des risques à propos du multimedia, de la voix, et de pas mal de nouveautés vraiment très belles. Je souhaite réellement qu'il le fassent.

Je ne tiens pas à comparer Microsoft à Apple. A l'origine, je me suis engagé dans cette voie, parce que je m'intéresse aux organisations et, pour moi, Apple était ce qu'il y avait de mieux en matière de structure. Au point où nous en sommes, j'aurais tendance à croire que c'est Microsoft.

Microsoft a eu l'avantage d'être là depuis longtemps. Nous avons fait plus de fautes que quiconque, et raté plus d'opportunités et de choses. Seulement voilà, nous voyons nos erreurs et nous en tirons les leçons. Je ne peux que m'en féliciter. Et puis il y a eu d'autres choses, telle l'interface graphique, où l'on a décidé de s'engager, et alors on s'y tient. Tout le monde pense que nous allons échouer « Mais pourquoi faites-vous ça ... c'est de la folie ». Nous, nous répondons « Nous avons intérêt à l'améliorer ».

Vous ne passez pas beaucoup de temps à penser à ce que souhaite le marché avant de prendre une nouvelle direction, n'est-ce pas ?

Non, vous ne pouvez pas dire cela. Ce n'est pas le seul terme de notre équation. On regarde ce qui se fait comme

matériels, et en comprenant la technologie qui y est attachée, on peut faire un pari ; que nous ne soyons pas encore en mesure de l'articuler aux utilisateurs finaux, ils peuvent le comprendre. Et puis il y a beaucoup d'autres choses, dont une forte proportion nous provient des retours d'information liés à la marche en avant de Windows. Ce sont ces choses qu'on appelle relation et insertion d'objets. En gros, les utilisateurs m'ont dit « S'il vous plaît, donnez-nous la mise en relation et l'encapsulage d'objets ! C'est une question primordiale pour notre compagnie d'assurance ! » (Rires). En fait c'est une façon de traduire l'idée des données mixtes - vous en avez vu un aperçu à la démonstration NeXT. Là où vous aviez certaines petites icônes et certains messages, vous pouviez « cliquer » dessus, et là était l'application. C'est la clé du principe de relation et d'insertion, de combinaison de ces choses, non seulement dans une messagerie, mais aussi dans unTableur ou un document de Traitement de Textes. Partout, vous pouvez avoir quelque chose et l'atteindre.

C'est ce qui, à mon avis, est un très grand progrès dans le système d'exploitation NeXT. Je suis content d'apprendre que cela sera intégré à vos produits.

Cela viendra avec des modifications dans l'application. Vous n'aurez même pas besoin d'une mise à jour du logiciel du système. C'est une chose que vous verrez beaucoup dans les applications Windows l'année prochaine. C'est chouette. (rires). Je pense l'avoir vu plus tôt que je ne m'y attendais !

Pour revenir à l'organisation de Microsoft, il semblerait que vous ayez réussi à y maintenir une composante humaine. La vieille forme d'organisation, inspirée des doctrines de Taylor, tendait à éliminer, autant que faire se pouvait, le contenu humain. Elles visaient à réduire chacun à un élément pouvant facilement être échangé ou remplacé. Je n'ai pas l'impression que vous dirigez Microsoft de cette façon.

Nous avons un grand nombre d'individus exceptionnels. La notion selon laquelle ce sont les architectes qui décident de leur façon de travailler, de contribuer à la vie de l'entreprise, s'adapte très bien aux individualités. La manière, dont je travaille avec le président, colle parfaitement à la façon dont j'aime passer le temps et exercer mes talents. Nos séminaires, nos réunions, tout cela convient aux différents groupes. Nous sommes confrontés à de nombreux défis, parce que nous avons grandi vite. Ce qui se fait facilement dans de petites entreprises devient délicat quand vous grossissez ; les gens disent « Ouais, vous avez un sacré avantage avec tous ces produits, qui font synergie...». Eh, si on retire ça, c'est énorme. De plus, les utilisateurs y trouveront leur avantage. C'est à coup sûr notre défi que chaque groupe de produits soit gagnant dans sa catégorie, de faire les choses qu'il faut, et de faire en sorte que les produits fonctionnent bien ensemble. Et je suis très content de notre façon de procéder, mais ce n'est pas simple.

J'ai passé beaucoup de temps sur le réseau, et j'ai le sentiment que ce à quoi nous devrions parvenir maintenant - le Grand Oeuvre - c'est la connexion des consciences. A ce jour, cet immense domaine des connexions installées est entièrement fondé sur UNIX. Voyez-vous Microsoft participer à cette activité?

Que voulez-vous dire ? Pensez-vous que les réseaux intra-entreprises soient fondés sur UNIX ? Pensez-vous qu'Arpanet soit à base d'UNIX ? Et puis quoi encore ?

Non, pas les réseaux intra-entreprises. Mais Usernet l'est, Internet l'est, les réseaux en développement, que j'ai essayés, le sont... Bon, je ne veux pas mettre UNIX sur un piédestal en tant que système d'exploitation, mais, là, il y a beaucoup d'UNIX.

En quel sens ? Je crois que vous faites des confusions, au plan technique. Les noeuds Usenet, vous ne pourriez pas savoir s'ils ne sont pas UNIX. Vous ne le sauriez jamais. Jamais. Envoyez un message Usenet à Microsoft et - [grattements de gorge] « Mon Dieu ! C'est un serveur OS/2 ! ». Désolé de vous le dire ! Cela vous paraît-il bizarre ? Qu'est-ce que UNIX ?

Alors, serait-ce pure coïncidence si tant de réseaux sont à base de machines UNIX ?

Ce n'est pas une coïncidence. Mais cela n'a rien à voir avec la technologie. Rien du tout. C'est une couche de protocoles. UNIX est un ensemble de fonctions de gestion de sémaphores et de mémoire.

Alors permettez-moi de reformuler ma question. Songez-vous à la manière pour populariser et étoffer la messagerie électronique et l'étendre à longues distances ?

Bien sûr. Il y a un certain nombre de choses sur lesquelles nous n'avons pas insisté, tant au niveau du système qu'à celui des applications. Dans la messagerie - et c'est vrai dans les deux cas - nous pouvons faire beaucoup. Nous avons déjà réalisé quelques petites choses - notre Mac Mail est un grand produit, il marche très bien, mais il n'a pas toute la structure ni l'étendue de la matière des communications, qui est considérable. Nous travaillons sur des projets passionnants. Nous parlons beaucoup avec les fabricants sur ce que signifierait que de disposer de télécopieurs intelligents se greffant sur le réseau, ou nous examinons quels éléments de la messagerie faudrait-il intégrer au logiciel du réseau. C'est le genre de choses qui doivent être fortement intégrées. Nous avons aussi d'autres domaines à explorer. Nous n'avons pas défini ce que serait une belle base de données graphique. Nous avons discuté de tout ce que nous devrions faire en matière de réseau. C'est une énorme somme de travail, il va de soi que Microsoft la fournisse.

NOTES

1. NdT. Nous aurions pu écrire « d'arrière en avant », mais le traducteur ne connaissant pas le personnage, il laisse au lecteur libre cours à sa sagacité.
2. NdT. contraction de « adolescent prolongé » (sic).
3. Million d'instructions Par Seconde.
4. Interface Graphique d'Utilisateur.
5. NdT. « flat data standards ».
6. NdT. du latin : « la façon de donner vaut mieux que ce qu'on donne ».
7. NdT. Interface de programmation d'applications.
8. NdT. Management Information Training Services. MITS était une société située à Albuquerque (Nouveau Mexique) pour laquelle Bill Gates a été consultant, et qui a disparu.
9. NdT. Club des Ordinateurs Bidouillés Maison.
10. NdT. nous en avons un exemplaire...
11. NdT. Sic, pour dire OS rapide et crasseux.
12. NdT. Independant Software Vendors.
13. NdT. Le goût prononcé des américains pour les acronymes et les mots composites nous a conduit à construire celui-ci pour coller avec « email » dans le texte. Ceux qui ont tâté de la terminologie absconse des nnnIX auront reconnu « electronic mail », d'où « melec » pour « messagerie électronique ».

De Pascal à C++

Le C++ est plus proche du langage Pascal que le C ne l'a jamais été. Il est donc temps d'y jeter un coup d'oeil !

par Jeff Duntemann
Traduction par Pierre Morel Fourrier

Il y a bien longtemps, à l'âge de l'Empire Funeste, j'ai facétieusement suggéré à mes amis hippies que la manière de gagner la guerre froide était tout simplement de naturaliser Américain tous les Russes. « Il est stupide d'utiliser des missiles nucléaires, ai-je remarqué, bombardons les plutôt de catalogues de chez Sears. Personne ne résiste à leurs soldes d'été ». Et en fin de compte, j'avais raison.

Je ne suis pas un adepte du langage C. Mais le C n'est pas une cause perdue. Il est très convenable si on l'utilise avec un peu de discipline et de structuration. Depuis des années, on me dit que Pascal et Modula-2 sont en train de perdre la guerre face au langage C, que le C triomphera et balaiera Pascal et Modula de la carte. Ce n'est pas si simple, quoique...

En fait, la guerre est un grand catalyseur du changement. Dans une bataille entre deux adversaires, même si l'un des deux peut gagner, les deux adversaires changent, en bien ou en mal. Alors si le langage C a jusqu'à maintenant gagné des batailles, il a aussi évolué. Cette évolution s'est faite en absorbant des caractéristiques d'autres langages, de la même manière que la culture américaine absorbe tranquillement et sans le moindre rejet tout ce qui passe à sa portée.

La dernière fois que j'ai jeté un oeil insistant sur le langage C, on en était déjà à la version 2.0 du C++. Alors j'ai eu un cri de joie : « On a gagné ! ».

Evidemment, vous n'avez pas besoin de répéter cela au programmeur C fanatique, mais le C++ 2.0 est plus proche du Pascal que d'un langage C millésimé 1980. La structuration et la discipline sont là. La plupart des bizarreries sont toujours là, mais tant pis. Il y a toujours des vendeurs de pop-corn en banlieue, mais vous n'êtes pas obligé d'aller là-bas et d'en acheter !

Mais pourquoi s'embêter ? Oui, pourquoi ? Turbo Pascal est un langage sacrément puissant, merci. Il possède toutes les caractéristiques d'un langage structuré (y compris les objets), et toutes les extensions spécifiques que l'on peut désirer. Si vous êtes déjà au paradis, pourquoi vouloir visiter le purgatoire ?

Voici une poignée de raisons que vous pourrez éventuellement trouver indiscutables :

- C++ permet une programmation orientée objets plus polyvalente que Turbo Pascal. Je suis un adepte de l'héritage multiple. C++ le permet mais pas Turbo Pascal, les constructeurs et les destructeurs de C++ sont considérablement plus puissants que ceux de Turbo Pascal, et le C++ peut contrôler plus facilement les droits d'accès aux objets.

- La portabilité est redevenu une réalité. Les différents environnements convergent maintenant, alors qu'ils ont divergé ces quinze dernières années. Le modèle d'interface utilisateur à base de fenêtres et souris créé par Xerox est maintenant en train de devenir universel. Quand il était absurde de porter une application écrite pour Macintosh vers un PC sous DOS, il devient maintenant raisonnable (à défaut d'être facile) de porter une application Macintosh vers Windows 3.0, vers X/Windows sous Unix, ou vers OS/2 Presentation Manager. C++ 2.0 est d'ores et déjà disponible sous ces environnements, ou du moins, le sera très bientôt.

- Il y a de quoi travailler en C++. Cela a d'ailleurs toujours été vrai du langage C pour ceux qui en avaient le courage. C++ est un langage à la fois plus difficile et plus facile à apprendre et à utiliser, selon que vous apprenez ce langage en profondeur ou non. Il m'est arrivé de penser que C++ peut s'étendre à la fois vers le haut - les sphères de la programmation scientifique et système - et vers le bas - les applications COBOLesques de lecture-de-fichier et d'écriture-d'enregistrement que la plupart d'entre nous ont eu à écrire un jour ou l'autre pour pouvoir

manger. Vraiment, C++ est ce que le langage ADA aurait dû être et ne sera jamais, et je peux quasiment vous promettre qu'il y aura toujours des emplois pour les programmeurs C++.

Une vue en hauteur

Informellement, un programme en Turbo Pascal est constitué de définitions de variables et de constantes au début du programme, des routines au milieu, et du programme principal encadré par BEGIN... END à la fin. Une des raisons pour lesquelles le C++ est facile à apprendre pour les programmeurs Pascal est qu'un programme C++ est généralement constitué de la même manière : les définitions au début, les routines au milieu, et le programme principal à la fin, même s'il n'est pas obligatoire de respecter cet ordre.

La figure 1 présente deux programmes côte-à-côte. Fonctionnellement, ils sont identiques, c'est-à-dire (et heureusement pour la longueur de cet article) qu'ils ne font pas grand chose. Ils sont juste là pour nous aider à faire notre voyage dans la structure du C++.

Le programme principal

Même pour les moins anglicistes d'entre nous, le C++ ne permet aucune erreur de localisation du programme principal. Il débute par main() (principal en anglais). Ce nom identifie un bloc de lignes comme étant le programme principal, et deux accolades encadrent ces lignes de codes pour former le bloc main. Plus généralement, les accolades en C++ ont la même fonction que les mots réservés BEGIN et END du Turbo Pascal.

Les parenthèses après le mot main sont obligatoires, mais nous ne nous en occuperons pas trop pour l'instant. Signalons seulement que main est exactement construit de la même manière qu'une fonction C++ quelconque. Et comme en C++, les paramètres passés à une fonction sont entre parenthèses, il y a donc des parenthèses après le nom de la fonction, même si - dans notre exemple - la fonction main n'a pas de paramètres.

Tous les sous-programmes en C++ sont appelés « fonctions », qu'ils retournent une valeur ou non. C'est pourquoi l'on entend parler de « fonction main », entendez « programme principal ». Le mot « void » (littéralement « vide ») signifie que la fonction main ne retourne aucune valeur. En C++, le mot void représente un type de données « vide », dans des situations où aucune information ne change de main.

Les programmes C++ n'ont pas de nom !

Le mot main, bien que n'étant pas réservé en C++, n'en est pas moins important. Ce mot marque le point d'entrée du programme, là où l'exécution commence.

Notez qu'il n'existe pas d'instruction au début d'un programme C++ lui donnant un nom. main n'est pas non plus le nom du programme. Les programmes C++ n'ont pas de nom, à la différence de leurs homologues en Pascal.

Enfin, il n'y a pas de caractère de ponctuation marquant la fin du programme alors qu'en Turbo Pascal, le END final doit être suivi d'un point.

Les commentaires

Il existe deux manières de noter un commentaire en C++. L'une des deux manières est similaire à celle du Pascal : tout ce qui se trouve entre une marque d'ouverture de commentaire /* et une marque de fin de commentaire */ est ignoré par le compilateur. Les marques /* et */ correspondent exactement aux (* et *) du Pascal.

L'autre manière ressemble aux commentaires de l'assembleur : une marque indique le début du commentaire, et la fin de la ligne après cette marque est ignorée par le compilateur. En C++, cette marque est //. Le compilateur C++ ignore la fin de la ligne après ces deux caractères.

Les définitions de variables

En C++, les variables sont définies « à l'envers », si l'on prend comme endroit la manière de déclarer une variable du Pascal. En effet, le type de la variable doit être spécifié avant le nom de la variable, et non après comme en Pascal. Autre différence par rapport au Pascal, aucun caractère de ponctuation ne doit être placé entre le type et le nom de la variable. Ainsi par exemple, « int x; » déclare une variable x de type entier. Les variables locales à un bloc doivent être déclarées à l'intérieur de ce bloc, c'est-à-dire après l'accolade ouvrante, et avant la première instruction de ce bloc.

Les définitions de fonctions

Immédiatement après ou avant la fonction main se trouvent les définitions des autres fonctions, qui sont en fait les sous-programmes au sens du Pascal. Tous les sous-programmes sont appelés fonctions en C++, qu'ils retournent ou non une valeur. En effet, en C++ on ne dit pas « procédure » à propos d'un sous programme qui ne retourne pas de valeur, mais fonction de type void.

Une définition de fonction est composée d'un en-tête et d'un bloc de lignes de code encadrées par des accolades. Ce code entre accolades est exécuté chaque fois que la fonction est appelée. Si la fonction possède des variables locales, celles-ci doivent être déclarées à l'intérieur du bloc, c'est-à-dire après l'accolade ouvrante et avant la première instruction. Ceci est différent du Pascal où les variables locales sont définies avant le BEGIN.

L'en-tête d'une fonction C++ est similaire à celui d'une fonction Pascal. Il contient le nom de la fonction ainsi que les types, noms et ordre de ses paramètres. L'en-tête indique aussi le type de la valeur retournée par la fonction. Si aucune valeur n'est retournée, le mot-clé réservé void doit être placé avant le nom de la fonction.

Les paramètres d'une fonction C++ sont déclarés comme les autres variables : le type en premier, puis le nom de la variable. Si une fonction possède plusieurs paramètres, ceux-ci sont séparés par des virgules, alors que Pascal les sépare par des points-virgule.

Par défaut, les paramètres sont transmis à la fonction par valeur, c'est à dire qu'une copie de la valeur de la variable est transmise à la fonction. C'est exactement identique à ce qui se passe en Pascal. A moins d'être précédé du mot VAR, un paramètre Pascal est transmis par valeur au sous programme. Le C++ permet aussi de transmettre des paramètres par référence, de manière très proche de ce qui se passe en Pascal. Un paramètre par référence n'est pas vraiment transmis à la fonction. En fait, seule l'adresse du para-

mètre est transmise, et les modifications faites sur un paramètre transmis par référence modifieront la valeur de la variable dans la fonction appelante. Encore une fois, les paramètres de type VAR du Pascal fonctionnent exactement de la même manière.

Le passage d'un paramètre par référence est indiqué en faisant précéder le nom du paramètre par le caractère & dans l'en-tête de la fonction :

```
void swap(int &x, int &y)
{
int temp;          // Définit une variable locale de type entier
  temp = x;        // Copie la valeur de x dans temp
  x = y;           // Copie la valeur de y dans x
  y = temp;        // Copie la valeur de temp dans y
}
```

Chaque paramètre doit posséder son propre indicateur de type, et son propre indicateur de passage par référence.

Les fonctions C++ ne peuvent être imbriquées

C'est une différence importante entre les sous programmes C++ et Turbo Pascal. Les fonctions C++ ne peuvent être imbriquées. En d'autres mots, il est impossible de déclarer une fonction à l'intérieur d'une autre. En Pascal par contre, les fonctions et les procédures peuvent contenir des fonctions et des procédures locales.

Le prototypage des fonctions

Le C++ impose une contrainte supplémentaire lors de la déclaration d'une fonction. Avant qu'une fonction ne soit définie ou appelée, l'en-tête de la fonction doit être recopié au début du module. Cet en-tête solitaire est nommé prototypage de la fonction. C'est en fait une bonne idée de rassembler les prototypages de toutes les fonctions au début de chaque programme.

L'ordre des fonctions

A la différence de Pascal, le C++ n'impose pas d'ordre particulier dans la définition des fonctions. En Pascal, vous devez éviter toute référence en avant en faisant attention de placer dans votre programme source les procédures appelées avant les procédures appelantes.

Le C++ connaît, du fait du prototypage, toutes les fonctions d'un programme. L'ordre des fonctions imposé par le Pascal n'est donc plus obligatoire.

Les fichiers header

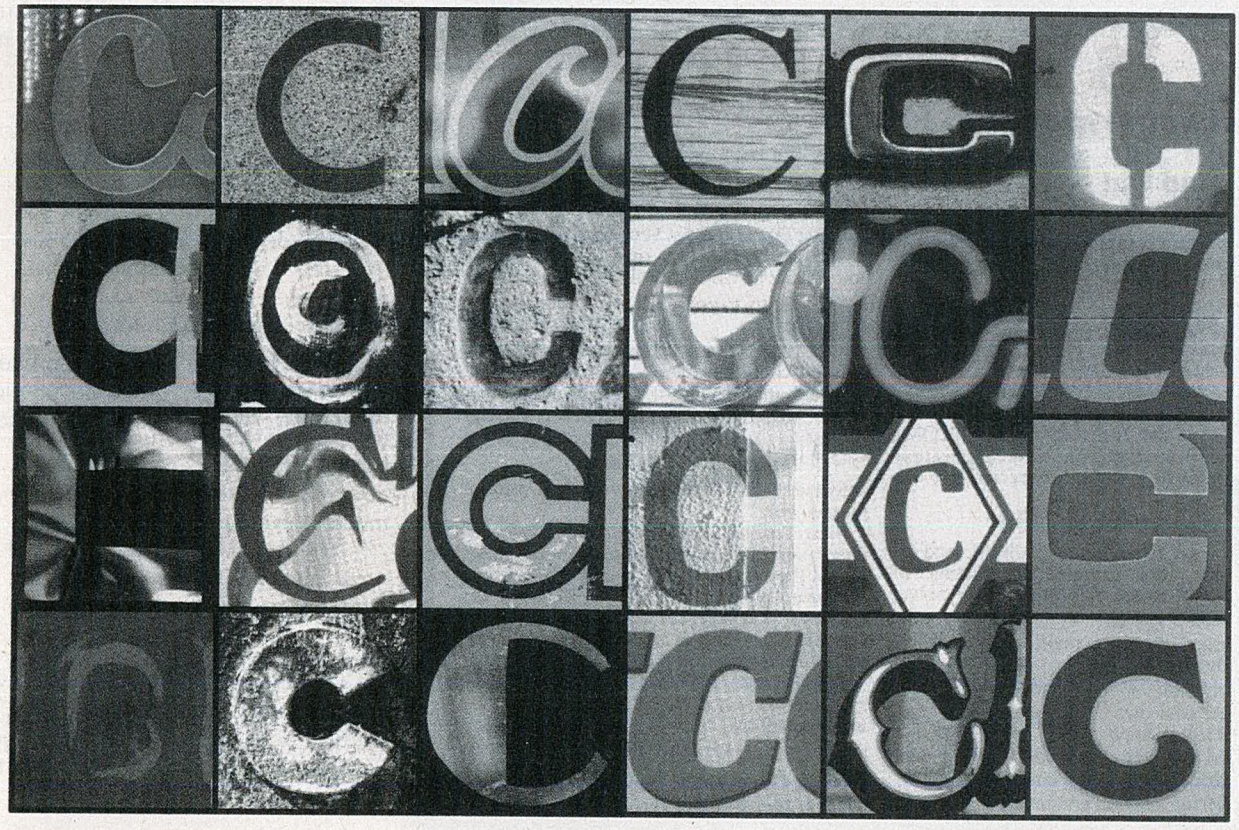
Vous verrez quasiment toujours au début d'un programme C++ des lignes du genre :

```
#include <stdio.h>
```

La plupart du temps figurent au moins deux lignes de ce style. La directive #include fait lire au compilateur le fichier source dont le nom est compris entre les caractères < et > comme s'il faisait partie du programme. Cela ressemble à la directive Turbo Pascal $I qui inclut un fichier source Pascal dans le fichier source en cours de compilation, même si l'usage en est généralement différent.

Les fichiers possédant une extension .H sont appelés fichiers Header. Ils sont fournis avec le compilateur (y compris le fichier stdio.h que l'on voit partout), même si vous pouvez en créer d'autres. Chargez-en un avec votre éditeur et examinez-le, mais surtout, ne le modifiez pas !

Vous y verrez des déclarations de constantes, de variables et de fonctions de la libraire. Plutôt que de com-



▲ LISTING 1 - BUBBLER.PAS

```
PROGRAM Bubbler:

CONST
   ARRAY_SIZE - 100:

TYPE
   IntArray - ARRAY[0...ARRAY_SIZE] OF INTEGER:

VAR
   I : Integer;
   TestArray  : IntArray;

PROCEDURE Swap (VAR X:Integer:VAR Y:Integer);

VAR Temp : Integer;

BEGIN
   Temp := X;  {Mémoriser la valeur de X dans le temps}
   X :=Y;      {Copier la valeur de l'y dans le x  }
   Y :=Temp;   {Copier la valeur du temps dans le y }

FUNCTION BubbleSort(VAR TargetArray:IntArray):Integer;

VAR SortedFlag, PassFlag. I : Integer;

BEGIN
   SortedFlag:=1;
  REPEAT
    PassFlag := 1;
    FOR I : =0 TO ARRAY_SIZE DO
     IF (TargetArray[I]>TargetArray[I+1]) THEN
        BEGIN
         Swap (TargetArray[I],TargetArray[I+1]),
         SortedFlag :=0;
         PassFlag   :=0;
        END;
  UNTIL PassFlag = 1;
  BubbleSort :=SortedFlag;
END;

PROCEDURE ShowArray (TargetArray : IntArray);

VAR I : Integer;

BEGIN
  FOR I : =0 TO ARRAY_SIZE DO
    BEGIN
      Write (TargetArray[I] : 3);
      IF ((I+1) MOD 20 = 0) THEN Writeln;
    END
   Writeln;
END;

BEGIN
  FOR I :=0 TO ARRAY_SIZE DO
   TestArray [i] :=Random (ARRAY_SIZE -1);
  Writeln ('Voici la table en désordre: ');
  ShowArray (TestArray);
  IF BubbleSort(TestArray) = 1
   THEN Writeln('Cette table a été triée!')
  ELSE
   BEGIN
     Writeln:
     Writeln('TLe tri de cette table a été terminé. La voilà:');
     ShowArray(TestArray);
   END;
END.
```

■ C++ et Turbo Pascal : une logique similaire

Voici l'exemple d'un même programme écrit en Turbo Pascal et en C++. Ce n'est pas un vrai programme; il crée un tableau de 100 entiers aléatoires; les affiche, les trie, et les affiche triés. Il existe d'autres manières d'écrire le même programme, mais celle-ci dépaysera très peu les programmeurs habitués à Turbo Pascal.

Ce que Turbo Pascal appelle paramètres VAR, C++ les appelle paramètres par référence, et les distingue en les faisant précéder d'un caractère &.

Les variables locales C++ sont déclarées à l'intérieur du bloc d'instructions. En Turbo Pascal, elles sont déclarées à l'extérieur du bloc, après le mot réservé VAR.

Le canal de sortie en C++ (représenté par l'opérateur <<) sur le périphérique cout est équivalent au Write du Pascal.

En C++, les variables globales doivent être déclarées dans le bloc main(). En Turbo Pascal, elle doivent être déclarées dans la section VAR au début du programme.

Le programme principal en Turbo Pascal doit être à la fin du programme. En C++, il peut être n'importe où, mais c'est une bonne habitude de le placer à la fin.

▲ LISTING 2 - BUBBLER.C

L'instruction C++ #include permet à un programme d'accéder à des librairies et à des fichiers de définitions. Elle a un peu le même rôle que l'instruction Pascal USES.

Dans un programme C++, toutes les fonctions doivent être prototypées, c'est-à-dire que tous les en-tête de fonctions doivent être spécifiés au début du programme. Cette section est similaire aux sections d'interface de Turbo Pascal.

Tous les sous-programmes C++ sont des fonctions, et sont précédées par le type retourné par la fonction. Une fonction C++ qui ne retourne pas de valeur doit être de type void.

Les fonctions C++ retournent une valeur en spécifiant la valeur à retourner après le mot réservé Return. En Turbo Pascal, la valeur à retourner doit être assignée au nom de la fonction.

Le bloc principal d'un programme C++ est toujours appelé main(). En Turbo Pascal, le bloc principal est simplement une série d'instructions encadrées par BEGIN et END.

En C++, un point-virgule termine une instruction, il ne les sépare pas. Cela signifie que le C++ nécessite des points-virgules là où Pascal n'en a pas besoin. Par exemple, juste avant une instruction else.

```
≠include<stdlib.h>
≠include<stdio.h>
≠include<iostream.h>
≠include<iomanip.h>

const int ARRAY_SIZE = 100;

void swap(int &x, int &y):
int bubble_sort (int*target_array);
void show_array(int*target_array);

void swap(int &x, int &y):
{
 int temp; //Définir la variable locale entière

 temp = x; //Mémoriser la valeur d'x dans le temps
 x - y;    //Copier l'y dans l'x
 y - temp; //Copier la valeur d'x de temps dans y
}

int bubble_sort(int*target_array)

{
 int sorted_flag = 1;
 int i.pass_flag;

do{
  pass_flag = 1;
  for (i=0; i <ARRAY_SIZE; i++)
    if (target_array [i]>target-array[i+1])
       {
       swap(target_array[i],target_array[i+1]);
       sorted_flag = 0;
       pass_flag   = 0;
       }
  }while (pass_flag = 0);

  return sorted_flag;
}
void show_array(int*target_array)
{
  int i,old_width:

  old-width = cout.width(5);
  for(i=o; i<100; i++)
    {
     cout <<setw(3)<<target_array[i];
     if ((i+1) % 20 =0) cout<<'\n';
    }
  cout<<'\n';
  cout.width(old_width);
}
void main ()
{
  int.i;
  int test_array[ARRAY_SIZE]://Déclarer une table de 100 entiers

  for(i=0; i<100; i++)
    test_array[i] = rand() % 100;
  cout <<"Voici la table en désordre:"<<'\n';
  show_array(test_array);
  if (bubble_sort(test_array))
    cout<<"Cette table a été déjà triée!";
  else
    cout<<"Le tri de cette table a été terminé. La voilà:\n";
    show_array(test_array);
  cout<<"\n";
}
```

pliquer le compilateur en y incluant la « prédéfinition » de ces valeurs (comme c'est le cas pour Turbo Pascal), ces définitions sont incluses dans des fichiers Header, et c'est au programmeur d'inclure dans ses programmes les fichiers dont il a besoin.

Quand vous commencerez à écrire des applications en C++, et à découper votre programme en plusieurs fichiers sources, vous trouverez très pratique d'inclure la définition de vos propres variables globales dans des fichiers de ce type. Vous serez sûr ainsi que les définitions sont identiques dans tous les modules !

Les points-virgules

Triste à dire, mais il existe en C++ des points-virgules comme en Pascal. Pire, les points-virgules servent en C++ à terminer une instruction, alors qu'en Pascal, ils servent à séparer deux intructions. En C++, chaque instruction doit être terminée par un point-virgule. Y compris la dernière instruction d'un bloc, alors qu'en Pascal une instruction avant un END n'a pas besoin d'être terminée par un point-virgule. De même lorsqu'une instruction se trouve juste avant un ELSE.

Un en-tête de fonction terminé par un point-virgule représente le prototypage d'une fonction. Mais, comme en Pascal, l'en-tête de la déclaration d'une fonction ne doit pas être terminé par un point-virgule.

Il vous sera donc facile d'oublier des point-virgules, mais rassurez vous, c'est moins douloureux que de renverser du thé bouillant sur vos genoux.

Le grand plongeon

Voici donc un résumé de l'anatomie des programmes C++, assez en tout cas pour vous mettre sur la ligne de départ. Vous passerez certainement beaucoup de temps à éplucher la documentation du compilateur, et à décortiquer les messages d'erreurs. Le C++ est un langage complexe et puissant, plein de subtilités.

Il y a deux manières de procéder. La manière classique est de commencer par écrire de petits programmes à partir de rien. C'est un peu comme entrer dans une piscine où l'eau est froide en avançant centimètre par centimètre. Cela peut durer très longtemps, et la manière brute (c'est celle que j'ai utilisée, vous vous en doutez) consiste à prendre l'un de vos programmes en Turbo Pascal, le charger avec votre éditeur, et de faire de la tête et des épaules jusqu'à ce qu'il se transforme en un programme C++. C'est très amusant et vous apprendrez énormément de chose en peu de temps.

Voici en gros le mode d'emploi de la traduction d'un programme Pascal en C++ :

1. Supprimer l'instruction PROGRAM au début du fichier, et remplacez la par l'instruction #include <stdio.h>.

2. Modifiez les commentaires. Vous pouvez pour cela utiliser la commande de recherche et remplacement de votre éditeur pour changer { et (* en */, puis } et *) en */.

3. Remplacez tous les BEGIN par {, et tous les END par }. Supprimez le point à la fin du programme.

4. Supprimez l'imbrication des procédures, en faisant attention à la portée des fonctions. Cela peut poser de gros problèmes, et si vous n'avez pas l'habitude d'imbriquer vos procédures en Turbo Pascal, la conversion sera beaucoup plus facile.

5. Transformez les déclarations de données en syntaxe C++. Si vous utilisez les objets de Turbo Pascal, passez un peu de temps à étudier les objets de C++ avant de vous jeter à l'eau. Heureusement, la transformation est quasiment immédiate. Mais transformer des objets C++ en objets Pascal est souvent impossible car les objets Pascal ne sont qu'un sous ensemble des objets du C++.

6. Déplacez les variables locales à l'intérieur du bloc de code des fonctions. Si vous imbriquez des procédures, vous devrez transformer certaines variables locales de la procédure qui contient les procédures imbriquées en variables globales. Et souvenez-vous que vos anciennes procédures locales deviennent des fonctions globales, et que les déclarations qui étaient locales à la procédure principale sont maintenant à la portée des anciennes procédures locales.

7. Transformez les paramètres des fonctions. Regardez la documentation avant de commencer. Cette transformation est immédiate.

8. Transformez la gestion de vos chaînes de caractères. L'octet de longueur de chaîne n'existe plus, et vous devrez aussi modifier les affichages à l'écran. Rappelez vous que vous ne pouvez utiliser l'opérateur égal pour assigner une chaîne de caractères. Utilisez à la place les fonction strcpy() ou strcat(). Il existe beaucoup de fonctions de gestion de chaînes de caractères dans la librairie, étudiez les.

9. Modifiez les instructions de contrôle en leur équivalent C++. Il existe un équivalent C++ pour chaque instruction du Pascal. Attention, il n'y a pas de THEN dans l'instruction IF du C++. Prenez garde à l'instruction Switch du C++ qui est beaucoup plus subtile que son équivalent Case du Pascal.

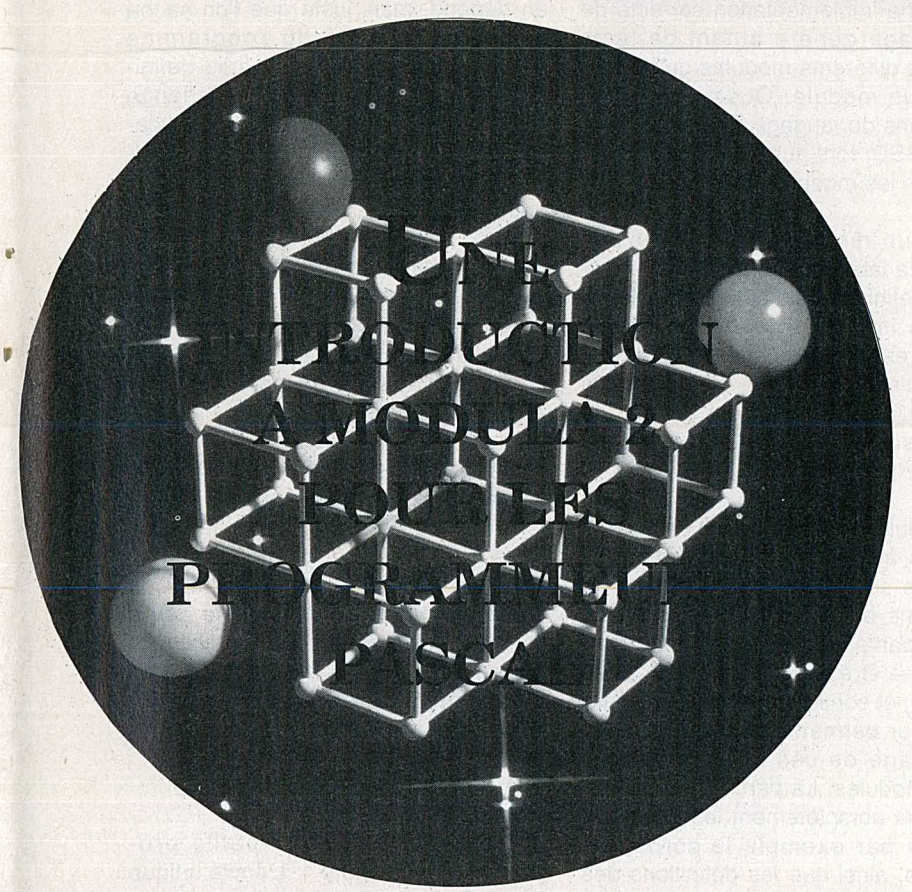
10. Essayez de trouver dans la librairie C++ les équivalents des fonctions de Turbo Pascal. Plus votre programme est complexe, plus cela sera difficile, et plus vous aurez de chance de devoir réécrire en C++ quelque-chose que vous pensiez acquis en Turbo Pascal.

En fait, si vous utilisez Turbo C++, vous serez en pays ami. Les deux langages Turbo utilisent la même librairie de fonctions BGI, et de nombreuses fonctions du Pascal ont un équivalent très proche en Turbo C++.

Réjouissez vous ! Vous êtes maintenant environ au tiers du chemin. Vous pouvez commencer à compiler votre programme. Evidemment, le résultat de la compilation vous laissera sans doute dans un état proche du désespoir, mais en traitant chaque bourde à la fois, cela finira par marcher !

Et une fois votre premier programme traduit en C++, prenez-en un autre... et répétez le processus jusqu'à ce que vous trouviez cela facile !

Cet article est extrait et adapté du livre de ***Jeff Duntemann,*** **« C++ for Turbo Pascal Programmers »,** à paraître en 1991.



Modula 2 est le nouveau langage créé par Niklaus Wirth, le père du langage Pascal. Qu'apporte-t'il de plus, et quelles difficultés peut éprouver un programmeur Pascal à apprendre Modula 2 ?

Par Hugh McLarty et David Smith
Traduction par Pierre Morel-Fourrier

Un programmeur Pascal peut apprendre Modula 2 en un après-midi, et il programmera confortablement dans ce nouveau langage en quelques jours. Dans ce but, nous présenterons donc Modula à un programmeur Pascal, en décrivant au fur et à mesure les différences et les similarités de ces deux langages.

Nous ne décrirons pas non plus l'intégralité de Modula 2. Les possibilités de ce langage concernant l'accès du matériel ou le multitâche sont hors du cadre de cet article. Signalons pour ceux qui désirent aller plus loin dans l'étude de ce langage le livre de Niklaus Wirth, « Programmer en Modula 2 ».

Une précision : nous nous référerons dans cet article, à la norme ISO du Pascal. Quasiment toutes les implémentations du langage Pascal présentent des extensions à cette norme, toutes incompatibles entre elles. En tout cas, peu ne possèdent pas toutes les spécificités de la norme ISO.

Les origines de Pascal

Si vous entendez l'expression « Programmation structurée », il y a de bonnes chances que l'on parle de Pascal. Même si l'on peut aussi penser au C, ADA ou même PL/1. Pascal, créé par Niklaus Wirth, est apparu dans la littérature en 1971. Il tire sa structure du langage ALGOL 60, créé en 1960, même si Wirth a travaillé sur d'autres langages entre 1960 et 1971, comme par exemple ALGOL-W, EULER ou PL360.

Wirth n'a pas été satisfait du langage ALGOL 68, version améliorée huit ans plus tard d'ALGOL 60, en partie parce qu'il était incompréhensible, même pour ses créateurs, et qu'il était très difficile de l'implémenter sur des ordinateurs moyennement puissants. PL/1 avait les mêmes problèmes.

Pascal, en revanche, a été créé dans le but d'être un langage simple, régulier et facile à comprendre. Ce langage est aussi étroitement associé à la fameuse (ou infâme) expression de « langage fortement typé », même si ce n'est pas le premier langage de ce genre.

Les origines de Modula 2

Les racines de Modula 2 se trouvent dans les langages Pascal, Modula et Mesa. Modula (c'est-à-dire Modula 1) a été une étape intermédiaire dans l'élaboration de Modula 2. Mesa est un langage qui a été créé par le centre de recherche Xerox à Palo Alto (PARC), et a été utilisé sur les ordinateurs Alto et Star.

En 1976, Wirth a passé une année sabbatique au PARC Xerox. Il a été impressionné par Mesa et par le fait qu'un seul langage de haut niveau pouvait être utilisé pour développer l'environnement complet d'un ordinateur personnel (système d'exploitation, compilateurs, éditeurs, etc...). Il retourna à l'Eidgenoessische Technische

Hochschule (ETH) en Suisse, et créa la machine Lilith, un micro ordinateur avec un écran graphique bitmap similaire à l'ordinateur Alto de Xerox. Tout l'environnement logiciel de Lilith a été écrit en Modula 2.

Comme Pascal, Modula 2 est un langage relativement simple. Il contient quelques structures logiques de contrôle simples. Il fournit de nombreux types de données comme les types énumérés, les ensembles, les enregistrements, et permet au programmeur de définir de nouveaux types. A la différence du Pascal, il propose des caractéristiques qui lui permettent l'écriture de gros programmes, le multitâche, et un accès de bas niveau au matériel.

Contrairement à Pascal, le langage Modula 2 ne définit pas de type de donnée concernant les fichiers ni d'instruction d'entrée sortie comme Get, Put, Read ou Write. En Modula 2, des modules, qui peuvent éventuellement être modifiés par le programmeur, fournissent les primitives d'accès aux fichiers. Le livre « Programmer en Modula 2 » décrit ces modules standards. Cette approche est d'ailleurs similaire au langage C qui fournit une librairie de fonctions pour gérer les entrées sorties.

Mais bien que les fonctions d'entrée sortie ne fassent pas partie des instructions du langage Modula, les développeurs de compilateurs Modula 2 se concertent pour fournir des librairies compatibles sur tous les compilateurs.

Modula 2 et la modularité

De nombreux dialectes de Pascal autorisent ce que l'on appelle la compilation séparée. Les différentes parties d'un programme sont compilées séparément, et assemblée à l'aide d'un éditeur de liens ou relieur. Mais si de nombreuses vérifications sont faites à l'intérieur d'un module pendant la compilation, très peu de tests sont effectués entre les différents modules. Par exemple, le nombre et le type des paramètres ne sont pas testés lors des appels de fonction hors d'un module. Il est même possible de définir une variable dans un module, et de s'en servir comme une procédure dans un autre module, avec, évidemment, des effets indésirables.

La définition du langage Modula 2 stipule que les modules sont compilés séparément, mais pas indépendamment. Une implémentation correcte de ce langage génère autant de tests entre les différents modules qu'à l'intérieur d'un module. Quelques implémentations du langage Pascal (notamment UCSD) font aussi des tests poussés entre les modules.

Quand un module doit utiliser des objets (variables, constantes ou procédures) définis dans d'autres modules, il doit les importer du module dans lequel il est défini en spécifiant le nom du module de définition, et le nom des objets. Un tel module qui utilise des objets sans les définir est appelé module client.

Les modules qui fournissent des objets destinés à être utilisés dans d'autres modules sont divisés en une partie définitions et une partie implémentation. La partie définitions liste les objets destinés à être utilisés dans les autres modules, et contient aussi des informations pour permettre des vérifications sur l'usage de ces objets dans les autres modules. La partie implémentation décrit complètement les objets, et contient par exemple le corps des fonctions, ainsi que les définitions des objets locaux au module.

Il est important de noter que de nombreuses sections d'implémentations différentes sont possibles pour une section de définitions donnée. Les détails d'une section d'implémentation sont inconnus des autres modules. Cette section se comporte donc comme une boîte noire, de la manière dont elle doit.

Cette technique est appelée masquage des informations, et c'est un moyen très puissant de structurer des systèmes. Une partie de sa puissance provient du fait que la plupart des erreurs de programmation mettent en cause les hypothèses de départ. Une erreur est souvent due à des hypothèses incorrectes. Le masquage des informations permet de localiser rapidement une information. Les informations auxquelles le programmeur n'a pas accès ne peuvent être en cause dans le cas d'une erreur. Et c'est un point fort de Modula 2.

Par exemple, lors d'un gros développement, une section d'implémentation simple et logiquement correcte est établie. Quand les interfaces du module sont fixées, il est possible de rendre plus complexe la partie implémentation. La modification de cette partie n'implique pas que l'on ait en tête la manière dont le reste du programme en dépend, mais juste que l'on sache comment le reste du programme dépend des objets de la partie définition. Si la partie définition ne change pas, on peut espérer que la modification de la partie implémentation n'induira pas d'erreur dans le reste du programme.

Prenons l'exemple d'une table des symboles comme on en utilise dans les compilateurs. On peut choisir une interface, et la formaliser en une partie définition. Ensuite on écrit la partie implémentation sous une forme simple, par exemple une liste linéaire de symboles. Cette liste linéaire est lente à utiliser, mais rapide et simple à mettre en oeuvre. Plus tard, on changera cette liste linéaire en une table de hashcode ou en un arbre binaire. Une partie définition bien écrite ne devra pas être modifiée lors de ces changements, et aucune modification ne devra être apportée aux autres modules du compilateur.

De Pascal vers Modula 2

Considérons les deux petits programmes de la figure 1. L'entité unique PROGRAM du Pascal est remplacée par le MODULE du Modula 2. Ce module est un exemple de module programme. D'autres types de modules

```
PROGRAM TestProgram (OUTPUT);
BEGIN
  WriteLn('Bonjour ');
END.
```

Figure 1a. **Programme simple en Pascal**

```
MODULE TestProgram;
  FROM InOutIMPORT
WriteString.WriteLn;
BEGIN
  WriteString("Bonjour");
  WriteLn
END TestProgram.
```

Figure 1b. **Programme simple en Modula-2**

sont décrits plus loin. Notez que les majuscules et les minuscules sont différenciées en Modula 2. En particulier, les mots-clé doivent tous être en majuscules. « WriteLn » est différent de « Writeln », et les deux sont différents de « WRITELN ».

Notre petit programme importe deux objets du module InOut. Ce sont deux procédures. Lors de la compilation, la partie définition du module InOut sera utilisée pour vérifier les appels de ce

```
TYPE DIRECTIONS = SET OF (up, down, left, right);
VAR barriers: DIRECTIONS;

...
barriers:=DIRECTIONS{up,left}
```

Figure 2. SET de Modula-2

module client. InOut est un module standard, décrit dans le livre « Programmer en Modula 2 ».

Les imports explicites demandent un petit effort d'écriture, mais indiquent rapidement au lecteur du programme quels objets sont utilisés, et d'où ils proviennent. Les seuls identificateurs qui sont implicitement importés sont les champs des enregistrements et les constantes énumérées.

Identificateurs, nombres et constantes

Le Pascal standard accepte des identificateurs de longueur quelconque et dont tous les caractères sont significatifs. La définition originale par Wirth et Jensen du langage Pascal spécifiait que seuls les huit premiers caractères étaient significatifs. De nombreuses implémentations du langage Pascal conservent encore ces anachronismes, particulièrement entre différents modules. La définition de Modula 2 ne dit rien sur la longueur des identificateurs, mais il semble largement répandu que les identificateurs peuvent être très longs, au moins 72 ou 80 caractères. Tous les caractères sont significatifs, et un identificateur trop long est considéré comme une erreur.

Les nombres réels comportent une légère différence. Un chiffre n'est pas obligatoire après le point décimal, mais ce point décimal est obligatoire. Par exemple :

5. est correct en Modula 2 mais pas en Pascal.
1E-3 est correct en Pascal mais pas en Modula 2.

Autre modification plus importante : Modula 2 autorise l'écriture de nombre en hexadécimal et en octal, ainsi que des constantes caractères dont le code Ascii est exprimé en octal :

0FFFFH Hexadécimal
0177564B Octal
15C (Constante caractère (Ascii CR)

Une chaîne de caractères en modula 2 est une suite de caractères encadrée soit par des guillemets, soit par des apostrophes. Une chaîne encadrée par des guillemets peut inclure des apostrophes, et vice-versa, mais il n'existe aucun moyen d'inclure à la fois des guillemets et des apostrophes dans la même chaîne.

Une chaîne de longueur n (n > 1) est du type :

ARRAY [0..n-1] OF CHAR

Une chaîne de caractères de longueur 1 est du type CHAR. Cela pose des problèmes car ce type n'est pas compatible avec le type ARRAY OF CHAR. Wirth est d'accord pour que le langage soit moins strict dans ce cas précis, et la prochaine version de Modula 2 pourrait inclure cette modification.

Types et déclarations

Un des principaux changements de Modula 2 par rapport au Pascal est qu'il peut y avoir autant de sections CONST, TYPE, VAR et PROCEDURE que l'on désire, et dans l'ordre que l'on désire. Il est aussi possible de réaliser des références en avant dans des instructions (par exemple à des types, constantes, variables et à d'autres procédures).

Un nouveau type numérique, CARDINAL, permet de travailler sur des entiers non signés. La valeur maximale d'un tel entier est MaxCard, et il est entendu que sur la plupart des machines qui permettent les calculs non signés, cette valeur est supérieure à MaxInt. La plupart des opérations possibles avec le type INTEGER le sont également avec le type CARDINAL.

Les types énumérés sont inchangés mais les sous-types sont maintenant encadrés par [et];

Les types SET sont les mêmes, les expressions sont légèrement différentes. Un type de set standard, SET, est défini de la manière suivantes :

TYPE BITSET = SET OF [0..n-1]

où n est la taille en bits d'un mot machine de l'ordinateur, généralement 16 ou 32 bits. BITSET fait partie intégrante du langage, mais il n'existe pas de valeur fixée pour n !

Les enregistrements ont une syntaxe légèrement différente, et il est possible d'avoir un nombre quelconque de parties variables (CASE) à n'importe quelle position (Figure 3). Modula 2 n'autorise pas l'utilisation des structures « Packed ».

```
RECORDx,y: REAL;
CASE tag: Color OF(*première partie variante *)
    red:a: CARDINAL      (* note, no (..) *)
   blue:b: INTEGER
  green:c: BITSET
  END; (*CASE*)
  z:[0..99];
  CASE BOOLEAN OF (*deuxième partie variante*)
    TRUE:    m,n: CARDINAL
    FALSE:   p: POINTER TO CHAR
  END (*CASE*)
END;(*RECORD*)
```

Figure 3. Expression RECORD DE Modula-2

Les procédures

Il existe deux différences mineures entre les procédures de Modula 2 et celles de Pascal. La première est qu'une liste de paramètres vide est autorisée, et évidemment obligatoire pour une fonction sans paramètres. La seconde est que le nom de la procédure doit être répété après le END qui termine la procédure.

En plus, une modification importante a été apportée dans l'utilisation des procédures. Modula 2 autorise la création de variables et de champ d'enregistrement de type procédure. On peut donc assigner une procédure à de telles variables. Et on appelle la procédure en question en spécifiant la variable de type procédure, suivie d'une liste de paramètres. La figure 4 illustre certaines de ces possibilités en définissant un tableau de procédures.

Dans le même domaine, Modula 2 résout un problème du Pascal concernant les tableaux passés en paramètre. En Modula 2, une fonction telle que celle décrite à la figure 5 est possible. La construction ARRAY OF signifie qu'un tableau de taille quelconque peut être transmis en paramètre à la fonction. A l'intérieur de la procédure, l'indice inférieur du tableau est zéro. L'indice maximal du tableau est donné par la fonction interne HIGH.

```
TYPE CommandProcedure = PROCEDURE(ch:CHAR);
VAR Command: ARRAY CHAR OF CommandProcedure;
    TimeToQuit: BOOLEAN;
PROCEDURE CommandInerpreter();
VARch: CHAR;
BEGIN
      TimeToQuit :=FALSE;
      REPEAT
            Terminal,Read(ch);(*lit un caractère au clavier*)
            Command[ch](ch);(*appelle la routine de commande *)
      UNTIL TimeToQuit;
END CommandInterpreter;
...
PROCEDURE InitializeCommandTable;
BEGIN
      FOR ch :=0C TO 177C DO
          Command[ch] :=IllegalCommand
      END;
      Command["q"] :=QuitCommand;
...
END InitializeCommandTable;
PROCEDURE IllegaleCommand(ch:CHAR);
BEGIN
      Terminal.Write(ch);Terminal.Write("?");
      Terminal.WriteLn;
ENDIllegalCommand;
PROCEDURE QuitCommand(ch:CHAR);
BEGIN
      TimeToQuit :=TRUE
END QuitCommand;
```

Figure 4. Exemple Modula 2, illustrations des procédures

Les expressions

Dans le domaine des expressions Modula 2, le programmeur Pascal rencontrera peu de surprises. Les mêmes opérateurs sont disponibles avec les mêmes priorités.

La seule surprise en stock pour les amoureux des expressions est une bonne surprise. Rappelons nous que la valeur du premier opérande d'un opérateur AND ou OR détermine s'il est nécessaire ou non d'évaluer le deuxième opérande pour obtenir la valeur de la condition entière. Par exemple, si le premier opérande d'un AND est faux, le résultat de l'expression est faux, quelle que soit la valeur du deuxième opérande. Dans ce cas, le programme Modula 2 n'évaluera pas le deuxième opérande en le « court-circuitant », contrairement à ce qui se passe en Pascal.

Si vous ne voyez pas l'importance que cela peut avoir, considérons le problème de la recherche d'un mot dans une liste, sachant que le mot peut ne pas être présent dans la liste. En Pascal, du fait que l'on ne sait pas si les deux opérateurs d'un AND sont évalués ou non, on doit écrire la boucle en faisant très attention à ne pas utiliser dans la condition de pointeur ayant la valeur NIL (Figure 6a). En Modula 2, on écrit la même boucle plus facilement (Figure 6b).

```
PROCEDURE Upcase(VARstr: ARRAY OF CHAR);
VAR i: CARDINAL;
BEGIN
    FOR i := 0 TO HIGH(str)DO
        str[i] := CAP(str[i])
    END
END Upcase;
```

Figure 5.Paramètres ARRAY en Modula 2

```
found := FALSE
WHILE (p<>NIL)AND NOT found DO BEGIN
   IF p^.next
       found := TRUE
   ELSE
       p := p^.next
END;
```

Figure 6a. Recherche linéaire en Pascal

```
WHILE(p<>NIL)AND(p^.key<>k)DO
   p := p^.next
END;
```

Fig 6b.Même fragment en Modula 2

Les instructions

L'une des premières choses que vous remarquerez en lisant des programmes Modula 2 est la relative rareté du mot-clé BEGIN. La plupart des instructions en Modula 2 incluent une terminaison (généralement END) et agissent sur une suite d'instruction, alors qu'en Pascal, les mêmes instructions agissent sur une seule instruction (même une si instruction unique peut être constituée d'une suite d'instructions encadrées par BEGIN END).

Modula 2 possède les mêmes structures de contrôle que Pascal, avec quelques améliorations, et une notable exception : le GOTO n'existe plus !

Comme on le voit à la figure 7, le WHILE DO est terminé par un END, et le REPEAT UNTIL est inchangé. En un certain sens, c'est l'instruction du langage Pascal la plus proche du Modula 2.

A cet ensemble d'instructions, Modula 2 ajoute la structure LOOP (figure 8) qui est exécutée jusqu'à ce qu'une instruction RETURN ou EXIT soit rencontrée. L'instruction EXIT n'est utilisable qu'à l'intérieur d'une boucle LOOP, et termine la dernière boucle LOOP rencontrée.

L'instruction FOR survit en Modula 2 avec une légère augmentation de sa puissance. Le DOWNTO est supprimé, et un BY le remplace pour spécifier le pas de chaque itération. Le pas par défaut est 1 bien sûr. L'instruction FOR est terminée par un END (Figure 9).

L'instruction CASE a subit le plus de changements. Notamment, une clause ELSE a été ajoutée pour traiter tous

```
WHILE(column<80)AND(Card[column] = "0")DO
     Card[column] := ' ';
     column := column + 1;
END
REPEAT
     p := p^.next;
     length := length + 1;
UNTIL p=NIL
```

Figure 7.Structure WHILE et REPEAT en Modula 2

```
LOOP
        ReadLine();
        IF eof EXIT;
        WriteLine();
END
```

Fig 8.Structure COOP en Modula 2

les cas non explicitement spécifiés. Cela était de toutes les façons présenté sous une forme ou une autre dans la plupart des implémentations de Pascal. Les valeurs possibles d'un CASE doivent toujours être des constantes, mais on peut spécifier des intervalles aussi bien que des valeurs uniques. Modula 2 autorise des expressions constantes alors que Pascal n'autorise que des valeurs simples. Les séquences d'instructions suivant le mot-clé CASE sont séparées par des barres verticales (Figure 11).

Les modules

L'unité en Modula 2 en terme de compilation est le module. En général, et comme expliqué plus haut, un module est découpé en deux parties. Une partie définition et une partie implémentation qui sont compilées séparément. La partie définition définit l'interface du module avec les autres modules : ce que fait le module. La partie implémentation (comment fonctionne le module) est intentionnellement cachée aux autres modules. Pour observer comment cela fonctionne, examinons l'exemple de la figure 12 qui met en oeuvre un système simple de pile.

En plus des fonctions push et pop (les deux fonctions de base), le module stack exporte deux variables pour signaler des éventuelles erreurs : Overflow (en cas de débordement de pile) et Underflow (la pile est vide). Un module peut exporter des procédures, des variables, des types ou des constantes.

Notez que les deux variables déclarées dans la section définition ne sont pas redéclarées dans la section implémentation. En fait, tous les types, variables ou constantes définies dans la section définition sont implicitement connues dans la section implémentation.

Le troisième sujet d'intérêt est le code à la fin de la section implémentation du module. Il est l'équivalent pour le module du programme principal d'un programme Pascal. Ce code est appelé le corps du module, et il peut être utilisé pour initialiser le module. Il est exécuté après que le corps de tous les modules importés soit exécuté. S'il y a des références circulaires entre les modules, l'ordre n'est pas défini.

Le module principal d'un programme Modula 2 est simplement un MODULE. Il peut (et c'est généralement le cas) importer des définitions, mais n'en exporte aucune.

Comment se passent les phases de compilation et d'édition de liens ? Comment Modula 2 s'assure-t'il que les bons modules sont bien utilisés ?

Chaque fois qu'une partie de définitions est compilée, un identificateur lui est assignée. Quand un autre module est compilé faisant référence à cette partie de définitions, l'identificateur est inclus dans le code compilé. L'implémentation de la partie de définitions et ses clients reçoivent cet identificateur.

Quand les modules sont reliés ensembles, tous les identificateurs de chaque section de définitions sont comparés. S'ils ne sont pas identiques, un message est généré. Cela survient quand deux modules sont compilés avec deux versions différentes d'une section de définitions, et qu'il peut donc y avoir des incompatibilités entre la déclaration des objets et leur utilisation.

Cette partie de l'article n'a pas été une description complète des modules. Nous renvoyons le lecteur intéressé au livre « Programmer en Modula 2 », particulièrement pour des renseignements complémentaires sur les modules ou la programmation de bas niveau.

Ceci n'est pas non plus l'histoire complète de Modula 2. Comme nous l'avons dit, nous n'avons pas parlé du multitâche ni de la programmation de bas niveau. Nous espérons seulement que le lecteur qui se posait des questions sur Modula 2 s'en pose désormais un peu moins, ou alors s'en pose de nouvelles.

```
FOR C := (Width - 1) TO Cursor BY - 1 DO
         Line[c + 1]:= Line [c];
END
```

Figure 9.Structure FOR en Modula 2

```
IF Total >2000.0 THEN
   IF Total >10000.0 THEN
       Total := Total + BigBonus
ELSE
      Total + SmallBonus;
```

Figure 10a.Indentation incorrecte en Pascal

```
 IF Total >2000.0 THEN
   IF Total >10000.0 THEN
       Total := Total + BigBonus
   ELSE
      Total + SmallBonus;
   END
END
```

Figure 10b. Même réécrit en Modula 2

```
CASE ch OF
   ' ':                Token := Blank
   '0'..'9':           Token := Number
   'A'..'F','a'..'z'   Token := Letter
               Token := Letter
ELSE Token := Unknown
END
```

Figure 11. Instruction CASE de Modula 2

```
DEFINITION MODULE IntStack;
       EXPORT Push,Pop,Overflow,Underflow;
       VAR Overflow,Underflow: BOOLEAN;
       PROCEDURE Push (Item : INTEGER);
       PROCEDURE POP (): INTEGER;
END IntStack.
IMPLEMENTATION MODULE IntStack;
       CONST StackSize = 100;
       VAR Stk : ARRAY[1..STACKSIZE]OF INTEGER;
             Top : [0..STACKSIZE];
       PROCEDURE Push (Item : INTEGER);
       BEGIN
              IF Top = StackSize THEN
                 Overflow :=TRUE;
              ELSE
                 Top := Top + 1;
                 Stk[Top] = Item;
              END
       END Push;
       PROCEDURE Pop () : INTEGER;
       BEGIN
              IF Top = 0 THEN
                 Underflow :=TRUE;
                 RETURN 0;
              ELSE
                 Top = Top - 1;
                 RETURN Stk[Top + 1];
       END Pop;
BEGIN (*Initialise la pile*)
       Top := 0;
       Overflow := FALSE;
       Underflow := FALSE;
END IntStack.
```

Figure 12. Mécanisme de la pile en Modula 2

Hugh McLarty est directeur du département programmation en Modula 2 de Logitech à Palo Alto. David Smith est employé chez Tolerant Systems.

UNE INTRODUCTION A MODULA 2 POUR LES PROGRAMMEURS PASCAL :
copyright **Dr. Dobb's Journal,** M&T Publishing Inc. Redwood City, CA USA 1990. All right reserved.

DISQUETTE LISTING DEBUG N° 11

1. KERMIT pour OS/2

2. PROTECT & VERIF nouvelles versions

3. GAG

Pour obtenir la disquette DEBUG 11, nous retourner le coupon détachable

DISQUETTES UTILITAIRES DEBUG MAGAZINE

utilitaires de compression et d'archivage

chaque disquette est vendue au prix de **47 Frs TTC**

DISQUETTE PKPAK
PCSIG n° 1330

DISQUETTE LHARC
PCSIG n° 2018

DISQUETTE LZE, LZESHELL
PCSIG n° 2316

DEBUG magazine

CAHIER LISTINGS

SOMMAIRE

11.1 : Kermit pour O/S 2 *(suite)* p 2

11.1 : KERMIT pour OS/2 *(suite)*

[LISTING QUATORZE]

```
DEFINITION MODULE KH;

CONST
   ID_OK         =  25;

   PARITY_OFF    =  150;
   ID_NONE       =  152;
   ID_ODD        =  151;
   ID_EVEN       =  150;

   STOP_OFF      =  140;
   ID_STOP2      =  142;
   ID_STOP1      =  141;

   DATA_OFF      =  130;
   ID_DATA8      =  138;
   ID_DATA7      =  137;

   BAUD_OFF      =  120;
   ID_B19K2      =  128;
   ID_B9600      =  127;
   ID_B4800      =  126;
   ID_B2400      =  125;
   ID_B1200      =  124;
   ID_B600       =  123;
   ID_B300       =  122;
   ID_B150       =  121;
   ID_B110       =  120;

   COM_OFF       =  100;
   ID_COM2       =  101;
   ID_COM1       =  100;

   IDM_C2        =  24;
   IDM_C1        =  23;
   IDM_AMBER     =  22;
   IDM_GREEN     =  21;
   IDM_WHITE     =  20;
   IDM_COLORS    =  19;
   IDM_DIREND    =  18;
   ID_DIRPATH    =  17;
   ID_SENDFN     =  16;
   IDM_DIRPATH   =  15;
   IDM_SENDFN    =  14;
   IDM_TERMHELP  =  13;
   IDM_HELPMENU  =  12;
   IDM_ABOUT     =  11;
   IDM_PARITY    =  10;
   IDM_STOPBITS  =  9;
   IDM_DATABITS  =  8;
   IDM_BAUDRATE  =  7;
   IDM_COMPORT   =  6;
   IDM_QUIT      =  5;
   IDM_REC       =  4;
   IDM_SEND      =  3;
   IDM_CONNECT   =  2;
   IDM_DIR       =  1;
   IDM_OPTIONS   =  52;
   IDM_FILE      =  51;
   IDM_KERMIT    =  50;

END KH.
```

[LISTING QUINZE]

```
IMPLEMENTATION MODULE KH;
END KH.
```

[LISTING SEIZE]

```
#define IDM_KERMIT      50
#define IDM_FILE        51
#define IDM_OPTIONS     52
#define IDM_HELP        0
#define IDM_DIR         1
#define IDM_CONNECT     2
#define IDM_SEND        3
#define IDM_REC         4
#define IDM_QUIT        5
#define IDM_COMPORT     6
#define IDM_BAUDRATE    7
#define IDM_DATABITS    8
#define IDM_STOPBITS    9
#define IDM_PARITY      10
#define IDM_ABOUT       11
#define IDM_HELPMENU    12
#define IDM_TERMHELP    13
#define IDM_SENDFN      14
#define IDM_DIRPATH     15
#define ID_SENDFN       16
#define ID_DIRPATH      17
#define IDM_DIREND      18
#define IDM_COLORS      19
#define IDM_WHITE       20
#define IDM_GREEN       21
#define IDM_AMBER       22
#define IDM_C1          23
#define IDM_C2          24
#define ID_OK           25
#define ID_COM1         100
#define ID_COM2         101
#define ID_B110         120
#define ID_B150         121
#define ID_B300         122
#define ID_B600         123
#define ID_B1200        124
#define ID_B2400        125
#define ID_B4800        126
#define ID_B9600        127
#define ID_B19K2        128
#define ID_DATA7        137
#define ID_DATA8        138
#define ID_STOP1        141
#define ID_STOP2        142
#define ID_EVEN         150
#define ID_ODD          151
#define ID_NONE         152
```

[LISTING DIX-SEPT]

```
IMPLEMENTATION MODULE DataLink;  (* Envoie et reçoit des paquets *)

   FROM ElapsedTime IMPORT
      StartTime, GetTime;

   FROM Screen IMPORT
      ClrScr, WriteString, WriteLn;

   FROM OS2DEF IMPORT
      HIWORD, LOWORD;

   FROM PMWIN IMPORT
      MPARAM, MPFROM2SHORT, WinPostMsg;

   FROM Shell IMPORT
      ChildFrameWindow, comport;

   FROM CommPort IMPORT
      CommStatus, GetChar, SendChar;

   FROM PAD IMPORT
      PacketType, yourNPAD, yourPADC, yourEOL;

   FROM KH IMPORT
      COM_OFF;

   FROM SYSTEM IMPORT
      BYTE;

   IMPORT ASCII;


   CONST
      MAXtime = 100;   (* en centièmes de secondes -> une seconde *)
      MAXsohtrys = 100;
      DL_BadCS = 1;
      DL_NoSOH = 2;


   TYPE
      SMALLSET = SET OF [0..7];   (* BYTE *)

   VAR
      ch : CHAR;
      status : CommStatus;


   PROCEDURE Delay (t : CARDINAL);
   (* delai en millisecondes *)

      VAR
         tmp : LONGINT;

      BEGIN
         tmp := t DIV 10;
         StartTime;
         WHILE GetTime() < tmp DO
         END;
      END Delay;


   PROCEDURE ByteAnd (a, b : BYTE) : BYTE;
      BEGIN
         RETURN BYTE (SMALLSET (a) * SMALLSET (b));
      END ByteAnd;
```

pliez ici en second et collez l'autre côté au ruban adhésif.

AFFRANCHIR
AU TARIF
LETTRE..

DEBUG MAGAZINE
3, rue de l'Atlas
75019 Paris

pliez ici en premier.

QUESTIONNAIRE – LECTEUR

Année de naissance : 19__
Sexe : ☐ M ☐ F

Votre secteur d'activité :

- ☐ Agriculture
- ☐ Enseignement
- ☐ Autres Fonction Publique, Collectivité locale
- ☐ Association
- ☐ Energie
- ☐ Métallurgie
- ☐ Chimie, verre, plastique
- ☐ Construction automobile, mécanique, navale, aéronautique
- ☐ Electronique
- ☐ Bâtiment, travaux publics
- ☐ Autres secteurs industriels
- ☐ Commerce
- ☐ Transports
- ☐ Télécommunications
- ☐ SSII
- ☐ Autres services aux entreprises
- ☐ Hôtellerie, réparation et autres services aux particuliers
- ☐ Banque, Assurance
- ☐ Autres secteurs commerciaux
- ☐ Informatique

Nombre de salariés dans l'établissement : ____

Votre profession :

- ☐ Propriétaire
- ☐ PDG-DG-Gérant
- ☐ Cadre
- ☐ Employé
- ☐ Ouvrier
- ☐ Retraité
- ☐ Autre ☐ Etudiant

Votre matériel :

- ☐ IBM ou compatible
- ☐ Apple, MAC
- ☐ Mini
- ☐ Mainframe

☐ Capacité de votre disque dur __ Mo
☐ Type de l'écran _____
☐ Valeur totale de votre installation _____ F TTC

L'utilisation de votre matériel :

- ☐ privée
- ☐ professionnelle
- ☐ les deux

Etes-vous satisfait de votre équipement actuel ? ☐ Oui ☐ Non
Allez-vous l'étendre ? ☐ Oui ☐ Non

Vos commentaires :

BULLETIN D'ABONNEMENT *2 ANS COMPLET*

❒ Je m'abonne pour **22** numéros + **22** disquettes listing + **2** numéros spéciaux : **990 F** au lieu de **2 002 F**
Je règle par : ❒ chèque à l'ordre de **Debug Magazine**
❒ carte bancaire
Nom de la carte : .....
N° ..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../ Date d'expiration /..../..../..../..../
Nom :..... Prénom : .....
Société :.....
Rue :.....
Lieu :.....
Code postal :..... Ville :.....
Pays :..... Tél : .....
Date :..... Signature : .....
L'abonnement commencera au n° .....

A renvoyer à **DEBUG MAGAZINE** - 3, rue de l'Atlas - 75019 Paris

(*) *Backup PRO* - offre valable dans les limites des stocks.

ET UN CADEAU DE BIENVENUE
le meilleur programme de sauvegarde du monde

-50 %

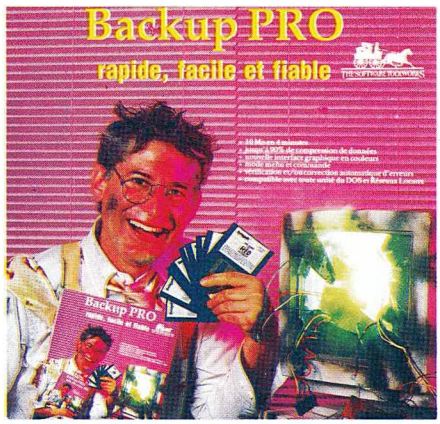
BULLETIN D'ABONNEMENT *2 ANS SIMPLE*

❒ Je m'abonne pour **22** numéros + **2** numéros spéciaux : **530 F** au lieu de **968 F**
Je règle par : ❒ chèque à l'ordre de **Debug Magazine**
❒ carte bancaire
Nom de la carte : .....
N° ..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../ Date d'expiration /..../..../..../..../
Nom :..... Prénom : .....
Société :.....
Rue :.....
Lieu :.....
Code postal :..... Ville :.....
Pays :..... Tél : .....
Date :..... Signature : .....
L'abonnement commencera au n° .....

A renvoyer à **DEBUG MAGAZINE** - 3, rue de l'Atlas - 75019 Paris

(*) *Backup PRO* - offre valable dans les limites des stocks.

ET UN CADEAU DE BIENVENUE
le meilleur programme de sauvegarde du monde



-45 %

BULLETIN D'ABONNEMENT *1 AN SIMPLE*

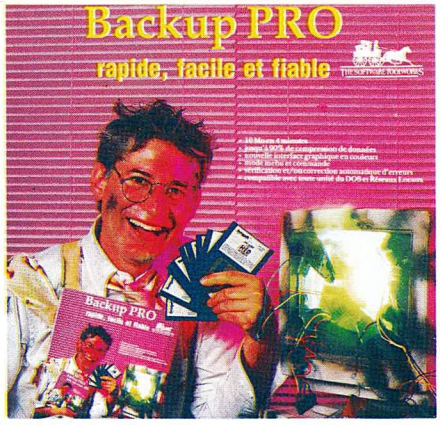
❒ Je m'abonne pour **11** numéros + **2** numéros spéciaux : **290 F** au lieu de **484 F**
Je règle par : ❒ chèque à l'ordre de **Debug Magazine**
❒ carte bancaire
Nom de la carte : .....
N° ..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../ Date d'expiration /..../..../..../..../
Nom :..... Prénom : .....
Société :.....
Rue :.....
Lieu :.....
Code postal :..... Ville :.....
Pays :..... Tél : .....
Date :..... Signature : .....
L'abonnement commencera au n° .....

Anciens numéros à **35 F** pièce : ❒ n° 2 ❒ n° 3 ❒ n° 4 ❒ n° 5 ❒ n° 6 ❒ n° 7 ❒ n° 8 ❒ n° 9
Anciens numéros à **37 F** pièce : ❒ n° 10

A renvoyer à **DEBUG MAGAZINE** - 3, rue de l'Atlas - 75019 Paris

(*) *Backup PRO* - offre valable dans les limites des stocks.

ET UN CADEAU DE BIENVENUE
le meilleur programme de sauvegarde du monde



-40 %

```
PROCEDURE Char (c : INTEGER) : CHAR;
(* convertit un nombre 0-95 en caractère imprimable *)
   BEGIN
      RETURN (CHR (CARDINAL (ABS (c) + 32)));
   END Char;


PROCEDURE UnChar (c : CHAR) : INTEGER;
(* convertit un caractère en nombre *)
   BEGIN
      RETURN (ABS (INTEGER (ORD (c)) - 32));
   END UnChar;


PROCEDURE FlushUART;
(* vide les registres "hold" de l'UART *)
   BEGIN
      Delay (500);
      REPEAT
         status := GetChar (comport - COM_OFF, ch);
      UNTIL status = NoCharacter;
   END FlushUART;


PROCEDURE SendPacket (s : PacketType);
(* Ajoute SOH et CheckSum à un paquet *)

   VAR
      i : CARDINAL;
      checksum : INTEGER;

   BEGIN
      Delay (10);   (* donne une chance à l'hôte de souffler *)
      FOR i := 1 TO yourNPAD DO
         status := SendChar (comport - COM_OFF, yourPADC, FALSE);
      END;
      status := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
      i := 1;
      checksum := 0;
      WHILE s[i] # 0C DO
         INC (checksum, ORD (s[i]));
         status := SendChar (comport - COM_OFF, s[i], FALSE);
         INC (i);
      END;
     checksum := checksum + (INTEGER (BITSET (checksum) * {7, 6}) DIV 64);
      checksum := INTEGER (BITSET (checksum) * {5, 4, 3, 2, 1, 0});
      status := SendChar (comport - COM_OFF, Char (checksum), FALSE);
      IF yourEOL # 0C THEN
         status := SendChar (comport - COM_OFF, yourEOL, FALSE);
      END;
   END SendPacket;


PROCEDURE ReceivePacket (VAR r : PacketType) : BOOLEAN;
(* supprime SOH et checksum — retourne status: TRUE = bon paquet *)
(* FALSE = timed out (pas de paquet) ou erreur de checksum *)

   VAR
      sohtrys : INTEGER;
      i, len : INTEGER;
      ch : CHAR;
      checksum : INTEGER;
      mycheck, yourcheck : CHAR;

   BEGIN
      sohtrys := MAXsohtrys;
      REPEAT
         StartTime;
         REPEAT
            status := GetChar (comport - COM_OFF, ch);
         UNTIL (status = Success) OR (GetTime() > MAXtime);
         ch := CHAR (ByteAnd (ch, 177C));   (* masque MSB *)
         (* ignore jusqu'à MAXsohtrys caractères de remplissage *)
         (* mais n'autorise que MAXsohtrys/10 timeouts          *)
         IF status = Success THEN
            DEC (sohtrys);
         ELSE
            DEC (sohtrys, 10);
         END;
      UNTIL (ch = ASCII.soh) OR (sohtrys <= 0);

      IF ch = ASCII.soh THEN
         (* reçoit le reste du paquet *)
         StartTime;
         REPEAT
            status := GetChar (comport - COM_OFF, ch);
         UNTIL (status = Success) OR (GetTime() > MAXtime);
         ch := CHAR (ByteAnd (ch, 177C));
         len := UnChar (ch);
         r[1] := ch;
         checksum := ORD (ch);
         i := 2;   (* deuxième caractère du paquet,
                      après l'octet de longueur *)
         REPEAT
            StartTime;
            REPEAT
               status := GetChar (comport - COM_OFF, ch);
            UNTIL (status = Success) OR (GetTime() > MAXtime);
            ch := CHAR (ByteAnd (ch, 177C));
            r[i] := ch;   INC (i);
            INC (checksum, (ORD (ch)));
         UNTIL (i > len);
         (* lit le caractère de checksum *)
         StartTime;
         REPEAT
            status := GetChar (comport - COM_OFF, ch);
         UNTIL (status = Success) OR (GetTime() > MAXtime);
         ch := CHAR (ByteAnd (ch, 177C));
         yourcheck := ch;
         r[i] := 0C;
         checksum := checksum +
                        (INTEGER (BITSET (checksum) * {7, 6}) DIV 64);
         checksum := INTEGER (BITSET (checksum) *  {5, 4, 3, 2, 1, 0});
         mycheck := Char (checksum);
         IF mycheck = yourcheck THEN   (* checksum OK *)
            RETURN TRUE;
         ELSE   (* ERREUR!!! *)
            WinPostMsg (ChildFrameWindow, WM_DL,
                        MPFROM2SHORT (DL_BadCS, 0), 0);
            RETURN FALSE;
         END;
      ELSE
         WinPostMsg (ChildFrameWindow, WM_DL,
                     MPFROM2SHORT (DL_NoSOH, 0), 0);
         RETURN FALSE;
      END;
   END ReceivePacket;


   PROCEDURE DoDLMsg (mp1, mp2 : MPARAM);
   (* Traite les messages DataLink *)
      BEGIN
         CASE LOWORD (mp1) OF
            DL_BadCS:
               WriteString ("Mauvais Checksum");   WriteLn;
          | DL_NoSOH:
               WriteString ("Pas de SOH");   WriteLn;
         ELSE
            (* Ne fait rien *)
         END;
      END DoDLMsg;

END DataLink.
```

[LISTING DIX-HUIT]

```
#include <os2.h>
#include "pckermit.h"

ICON IDM_KERMIT pckermit.ico

MENU IDM_KERMIT
   BEGIN
      SUBMENU "~Fichier", IDM_FILE
         BEGIN
            MENUITEM "Ré~pertoire...",          IDM_DIR
            MENUITEM "~Connexion\t^C",          IDM_CONNECT
            MENUITEM "~Envoi...\t^E",           IDM_SEND
            MENUITEM "~Réception...\t^R",       IDM_REC
            MENUITEM SEPARATOR
            MENUITEM "~Quitte\t^Q",             IDM_QUIT
            MENUITEM "~A propos de PCKermit...",IDM_ABOUT
         END

      SUBMENU "~Options", IDM_OPTIONS
         BEGIN
            MENUITEM "Port ~COM...",        IDM_COMPORT
            MENUITEM "Vitesse en ~Bauds...",IDM_BAUDRATE
            MENUITEM "Bits de ~Données...", IDM_DATABITS
            MENUITEM "Bits de ~Stop...",    IDM_STOPBITS
            MENUITEM "Bit de ~Parité...",   IDM_PARITY
         END

      SUBMENU "~Couleurs", IDM_COLORS
         BEGIN
            MENUITEM "Mono ~Blanc",     IDM_WHITE
            MENUITEM "Mono ~Vert",      IDM_GREEN
            MENUITEM "Mono ~Ambre",     IDM_AMBER
            MENUITEM "Couleurs ~1",     IDM_C1
            MENUITEM "Couleurs ~2",     IDM_C2
         END

      MENUITEM "F1=Aide",    IDM_HELP, MIS_HELP | MIS_BUTTONSEPARATOR
   END

ACCELTABLE IDM_KERMIT
   BEGIN
      "^C", IDM_CONNECT
      "^E", IDM_SEND
      "^R", IDM_REC
      "^Q", IDM_QUIT
   END

DLGTEMPLATE IDM_COMPORT LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_COMPORT, 129, 91, 143, 54, FS_NOBYTEALIGN | FS_DLGBORDER |
               WS_VISIBLE | WS_CLIPSIBLINGS | WS_SAVEBITS
    BEGIN
        CONTROL "Port COM :     ", IDM_COMPORT, 10, 9, 83, 38,
                WC_STATIC, SS_GROUPBOX | WS_VISIBLE
        CONTROL "COM1", ID_COM1, 30, 25, 43, 10, WC_BUTTON,
                BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_VISIBLE
        CONTROL "COM2", ID_COM2, 30, 15, 39, 10, WC_BUTTON,
                BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "OK", ID_OK, 101, 10, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                BS_DEFAULT | WS_GROUP | WS_TABSTOP | WS_VISIBLE
    END

END
```

```
DLGTEMPLATE IDM_BAUDRATE LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_BAUDRATE, 131, 54, 142, 115, FS_NOBYTEALIGN |
                FS_DLGBORDER | WS_VISIBLE | WS_CLIPSIBLINGS | WS_SAVEBITS
    BEGIN
        CONTROL "Vitesse en Bauds", IDM_BAUDRATE, 8, 6, 85, 107,
                WC_STATIC, SS_GROUPBOX | WS_VISIBLE
        CONTROL "110 Baud", ID_B110, 20, 90, 62, 10, WC_BUTTON,
                  BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_VISIBLE
        CONTROL "150 Baud", ID_B150, 20, 80, 57, 10, WC_BUTTON,
                  BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "300 Baud", ID_B300, 20, 70, 58, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "600 Baud", ID_B600, 20, 60, 54, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "1200 Baud", ID_B1200, 20, 50, 59, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "2400 Baud", ID_B2400, 20, 40, 63, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "4800 Baud", ID_B4800, 20, 30, 62, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "9600 Baud", ID_B9600, 20, 20, 59, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "19.200 Baud", ID_B19K2, 20, 10, 69, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "OK", ID_OK, 100, 8, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                    BS_DEFAULT | WS_GROUP | WS_TABSTOP | WS_VISIBLE
    END
END

DLGTEMPLATE IDM_DATABITS LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_DATABITS, 137, 80, 140, 56, FS_NOBYTEALIGN |
                FS_DLGBORDER | WS_VISIBLE | WS_SAVEBITS
    BEGIN
        CONTROL "Bits de donnée :", IDM_DATABITS, 8, 11, 80, 36,
                WC_STATIC, SS_GROUPBOX | WS_VISIBLE
        CONTROL "  7 bits   ", ID_DATA7, 15, 25, 67, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_VISIBLE
        CONTROL "  8 bits   ", ID_DATA8, 15, 15, 64, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "OK", ID_OK, 96, 12, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                    BS_DEFAULT | WS_GROUP | WS_TABSTOP | WS_VISIBLE
    END
END

DLGTEMPLATE IDM_STOPBITS LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_STOPBITS, 139, 92, 140, 43, FS_NOBYTEALIGN |
                FS_DLGBORDER | WS_VISIBLE | WS_SAVEBITS
    BEGIN
        CONTROL "   Stop bits    ", IDM_STOPBITS, 9, 6, 80, 32,
                WC_STATIC, SS_GROUPBOX | WS_VISIBLE
        CONTROL "1 Stop Bit", ID_STOP1, 20, 20, 57, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_VISIBLE
        CONTROL "2 Stop Bits", ID_STOP2, 20, 10, 60, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "OK", ID_OK, 96, 8, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                    BS_DEFAULT | WS_GROUP | WS_TABSTOP | WS_VISIBLE
    END
END

DLGTEMPLATE IDM_PARITY LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_PARITY, 138, 84, 134, 57, FS_NOBYTEALIGN | FS_DLGBORDER |
                WS_VISIBLE | WS_SAVEBITS
    BEGIN
        CONTROL "   Parité    ", IDM_PARITY, 12, 6, 64, 46, WC_STATIC,
                SS_GROUPBOX | WS_VISIBLE
        CONTROL "Pair", ID_EVEN, 25, 30, 40, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_VISIBLE
        CONTROL "Imp", ID_ODD, 25, 20, 38, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "Sans", ID_NONE, 25, 10, 40, 10, WC_BUTTON,
                    BS_AUTORADIOBUTTON | WS_TABSTOP | WS_VISIBLE
        CONTROL "OK", ID_OK, 88, 8, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                    BS_DEFAULT | WS_GROUP | WS_TABSTOP | WS_VISIBLE
    END
END


DLGTEMPLATE IDM_ABOUT LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_ABOUT, 93, 74, 229, 88, FS_NOBYTEALIGN | FS_DLGBORDER |
                WS_VISIBLE | WS_SAVEBITS
    BEGIN
     ICON IDM_KERMIT -1, 12, 64, 22, 16
     CONTROL "  PCKermit OS/2  ", 256, 67, 70, 82, 8, WC_STATIC,
                    SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "Copyright (c) 1990  Brian R. Anderson  ", 257, 27, 30, 172, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
    CONTROL "Communication de Micros vers Mainframes  ", 259, 13, 50, 199, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "  OK  ", 258, 88, 10, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                BS_DEFAULT | WS_TABSTOP | WS_VISIBLE
    END
END

DLGTEMPLATE IDM_HELPMENU LOADONCALL MOVEABLE DISCARDABLE
BEGIN
  DIALOG "", IDM_HELPMENU, 83, 45, 224, 125, FS_NOBYTEALIGN | FS_DLGBORDER |
                WS_VISIBLE | WS_CLIPSIBLINGS | WS_SAVEBITS

    BEGIN
    ICON IDM_KERMIT -1, 14, 99, 21, 16
    CONTROL "Menu d'aide Kermit", 256, 64, 106, 91, 8, WC_STATIC,
                SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
    CONTROL "Options de communication ..... Alt, O",
                258, 10, 80, 201, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
    CONTROL "Connexion à l'hôte ..... Alt, F; C",
                259, 10, 70, 204, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
   CONTROL "Répertoire ..... Alt, F; P",
                260, 10, 60, 207, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
    CONTROL "Envoie un fichier ..... Alt, F; E",
                261, 10, 50, 207, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
    CONTROL "Reçoit un fichier ..... Alt, F; R",
                262, 10, 40, 209, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
    CONTROL "Quitte ..... Alt, F; Q",
                263, 10, 30, 205, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
    CONTROL "OK", 264, 83, 9, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                    WS_TABSTOP | WS_VISIBLE | BS_DEFAULT
  END
END

DLGTEMPLATE IDM_TERMHELP LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_TERMHELP, 81, 20, 238, 177, FS_NOBYTEALIGN |
                FS_DLGBORDER | WS_VISIBLE | WS_CLIPSIBLINGS | WS_SAVEBITS
    BEGIN
     CONTROL "^E = Mode Echo", 256, 10, 160, 72, 8, WC_STATIC,
                SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "^L = Mode echo local", 257, 10, 150, 97, 8, WC_STATIC,
                SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "^T = Mode Terminal sans echo", 258, 10, 140, 131, 8, WC_STATIC,
                SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "^N = Mode conversion <cr> -> <cr><lf>", 259, 10, 130, 165, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "^O = Sans conversion ", 260, 10, 120, 109, 8, WC_STATIC,
                SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL " Emulation terminaux TVI950 / IBM 7171", 261, 10, 100, 217, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "Sh-F1 - Sh-F12   =   PF1 - PF12", 262, 10, 90, 135, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "Home                =  Eff. ", 263, 10, 80, 119, 8, WC_STATIC,
                SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
     CONTROL "PgDn                 =  Page  Bas                ",
                264, 10, 70, 228, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
     CONTROL "PgUp                 =  Page Haut                ",
                265, 10, 60, 227, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
        CONTROL "Insert   =  Mode insertion      ", 266, 10, 40, 221, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
        CONTROL "Delete               =  Efface", 267, 10, 30, 199, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
        CONTROL "Control-G   =  Réaffiche écran      ", 268, 10, 20, 222, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
        CONTROL "Touches curseur     = Fonctions habituelles        ",
                269, 10, 10, 229, 8, WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP |
                WS_GROUP | WS_VISIBLE
        CONTROL "OK", 270, 193, 158, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                BS_DEFAULT | WS_TABSTOP | WS_VISIBLE
        CONTROL "End                  =  Fin       ", 271, 10, 50, 209, 8,
                WC_STATIC, SS_TEXT | DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
    END
END


DLGTEMPLATE IDM_SENDFN LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_SENDFN, 113, 90, 202, 60, FS_NOBYTEALIGN | FS_DLGBORDER |
                WS_VISIBLE | WS_SAVEBITS
    BEGIN
        CONTROL "Envoi :  ", 256, 4, 4, 195, 24, WC_STATIC, SS_GROUPBOX |
                WS_GROUP | WS_VISIBLE
        CONTROL "Nom de fichier:", 257, 13, 11, 69, 8, WC_STATIC, SS_TEXT |
                DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
          ICON      IDM_KERMIT -1, 15, 38, 22, 16
        CONTROL "PCKermit  OS/2   ", 259, 59, 45, 82, 8, WC_STATIC, SS_TEXT |
                DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
        CONTROL "OK", 260, 154, 36, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                WS_TABSTOP | WS_VISIBLE | BS_DEFAULT
        CONTROL "", ID_SENDFN, 89, 10, 98, 8, WC_ENTRYFIELD, ES_LEFT |
                    ES_MARGIN | WS_TABSTOP | WS_VISIBLE
    END
END

DLGTEMPLATE IDM_DIRPATH LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_DIRPATH, 83, 95, 242, 46, FS_NOBYTEALIGN | FS_DLGBORDER |
                WS_VISIBLE | WS_SAVEBITS
    BEGIN
        CONTROL "Directory", 256, 7, 5, 227, 24, WC_STATIC, SS_GROUPBOX |
                WS_GROUP | WS_VISIBLE
        CONTROL "Path:", 257, 28, 11, 26, 8, WC_STATIC, SS_TEXT | DT_LEFT |
                DT_TOP | WS_GROUP | WS_VISIBLE
        CONTROL "OK", 258, 185, 31, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                WS_TABSTOP | WS_VISIBLE | BS_DEFAULT
        CONTROL "*.*", ID_DIRPATH, 57, 11, 166, 8, WC_ENTRYFIELD, ES_LEFT |
                    ES_AUTOSCROLL | ES_MARGIN | WS_TABSTOP | WS_VISIBLE


  END
END
```

à découper

LOGONE

Siège social : 36, cours du Chapeau Rouge 33000 BORDEAUX- Tél. 56 44 21 11 Fax. 56 51 73 94
Direction commerciale : 14, rue Plaisance 75014 PARIS Tél. 45 39 69 18 Fax. 45 41 47 96

n° 3
AVRIL 91

NEWS

EDITO

Certains logiciels se comportent différemment, selon qu'ils sont francisés ou non, la référence de version restant la même. Quelques uns sont moins complets dans leur version française.

C'est pourquoi nous vous en proposons certains en V.O. Un manuel en français, rédigé par notre équipe d'informaticiens linguistes, complète leur documentation, le cas échéant.

Comme dans ses éditions précédentes, ce bulletin vous donne des informations sur les nouveaux produits et les promotions de la saison commerciale.

Trois nouvelles rubriques sont créées :

FINESSE Nous y publions les idées et astuces émanant de vos observations, découvertes ou difficultés résolues.

SYNERGIE La judicieuse collaboration d'outils hétérogènes pour créer une activité ou en améliorer les performances.

COUP D'ŒIL On y présente les caractéristiques originales de produits apportant de solides solutions à des problèmes répandus.

et un nouveau service :

FOUINEUR Appelez-nous, dites-nous ce que vous voudriez trouver, et nous nous mettrons en chasse ! Les bonnes relations que nous avons établies sur les marchés européen et nord-américain sont là pour ça ! ■

NOUVEAUTES

MICROSOFT

Nous ouvrons notre gamme tenue en stocks aux produits de la ligne WINDOWS, à commencer par Windows 3 lui-même. Pour un service plus complet, nous avons sélectionné des logiciels originaux pour Windows.

101 - WINDOWS

Collection de 101 programmes pour Windows, choisis parmi les plus puissants, les plus faciles d'utilisation et les plus récents de la bibliothèque internationale. Quelle que soit leur origine, ils sont tous en V.O.

PACK ANTIVIRUS pour Windows 3

Ce logiciel identifie plus de 200 "virus". Entièrement paramétrable, il bénéficie de tout le confort dû à l'environnement Windows 3.

OMNIPAGE/386 v. 3.0 pour Windows 3

Mêlés d'images, disposés en colonnes, rédigés en l'une ou l'autre de 11 langues européennes, vos textes sont enfin lus avec fiabilité via votre scanner ! Il n'y a plus matière à jongler avec les alphabets électroniques nationaux ! Pas de problème non plus avec les tailles des fontes : OMNIPAGE lit les tailles de 6 à 72 points. Ni avec les caractères proportionnels. Ni avec les attributs (soulignement, graisse, italique). OMNIPAGE gère les formats portrait et paysage, tient compte des décalages, justifications et centrages, colonnes et images. Enfin, il sait traiter les fichiers Fax reçus. Compatible avec pratiquement tous les logiciels,

(Suite page 2)

NOUVEAUTES

(suite de la page 1)

OMNIPAGE permet de produire un fichier exploitable au format de votre traitement de texte préféré. Sa fiabilité et ses performances en font un remarquable outil de productivité.

HUGO PLUS

Correcteur syntaxique et lexical, HUGO PLUS fait partie des prix d'excellence attribués par SVM n° 79 pour l'année 90. HUGO PLUS gère plus de 400.000 formes avec toutes leurs combinaisons. L'utilisateur peut créer et charger jusqu'à 8 dictionnaires spécialisés. Outre les fichiers ASCII, HUGO PLUS traite les fichiers WordStar 4, Word 5, etc. Il procède par suggestion, tant pour corriger que pour enregistrer. Un éditeur de texte, simple et bien fait, y est incorporé.

FRENCH ASSISTANT

Selon qu'on le charge en mode résident ou comme un programme traditionnel, FRENCH ASSISTANT se comporte comme un dictionnaire bilingue (Anglais-Français/Français-Anglais) intervenant directement dans le texte affiché à l'écran, quel que soit le traitement de texte utilisé, sur appel de touches composées ; ou bien il traite un fichier ASCII et en propose, phrase par phrase, la traduction. Pour certains mots, il sait en offrir les variantes contextuelles. Ses dictionnaires peuvent être modifiés et complétés. Plus qu'un confort, FRENCH ASSISTANT est un outil de productivité. En tant que traducteur, il fonctionne dans le sens Anglais-Français. En tant que dictionnaire bilingue résident, il fonctionne dans les deux sens.

ARTLINE 2

La lettre est un art et le document un dessein !

Ce puissant outil de P.A.O. offre le meilleur rapport qualité/prix du marché. Ses riches fonctions graphiques (fonctions de Bezier) permettent de produire des documents élégants et percutants. ARTLINE 2 donne des ailes à vos concepteurs graphiques. Donnez-lui un 386, 4 MO de mémoire (à moins, ça marche, mais on est limité en "clipart") et une bonne imprimante laser.

LASERTWIN

Parce que beaucoup de programmes graphiques, de traitements de textes et de P.A.O. sont dédiés aux imprimantes HP LaserJet + et Série II, vous ne pouvez les exploiter convenablement avec votre Canon LBP8 ou LBP4. Ce problème est résolu par LASERTWIN : fonctionnant en mode résident, il filtre les commandes en mode HP et les convertit en commandes idoines pour votre Canon. C'est la solution idéale pour exploiter pleinement votre imprimante laser Canon.

SUPERFONT 25/1

C'est l'alternative aux cartouches coûteuses et peu pratiques. Avec SUPERFONTS 25/1 toutes les polices des cartouches HP A à Z sont disponibles directement à partir du disque. Vous n'avez qu'à sélectionner, à partir d'un menu, les fontes que vous désirez utiliser.

SUPERFONTS fonctionne avec toutes les imprimantes HP et compatibles ainsi qu'avec les logiciels les supportant. Combinez SUPERFONTS 25/1 avec LASERTWIN pour utiliser une imprimante Canon.

OVERCOM

Logiciel de télécommunication multiprotocole (Xmodem, Ymodem, Kermit, etc.). Téléchargez des fichiers à partir de serveurs tels que GOTO, SM1, VIF, etc. Emulation Minitel, VT52, VT100 et autres terminaux. Fonctionne en mode commande ou par menus déroulants. Enregistre les informations consultées en fichiers ASCII ou Videotex.

MEMOTEL II

Logiciel d'exploitation de l'annuaire électronique. Compatible avec le modem des Minitel (du M1 au M10), il permet de récupérer les adresses sur critères. MEMOTEL II est aussi compatible avec la plupart des cartes modem Videotex (LCE-TEL, KX-TEL, compatibles Hayes, etc.). Permet de créer des fichiers de procédure d'interrogation. Traitement de codes postaux. Fonction d'édition paramétrable d'étiquettes à partir des fichiers d'adresses récupérées. MEMOTEL II est un outil de prospection commerciale. □

SYNERGIE

Réaliser un atelier de traduction productif

Vous recevez un manuel en Anglais. Il vous faut en produire rapidement la version française. Au moyen d'**OMNIPAGE**, sous **WINDOWS** 3, et d'un scanner à plat, vous allez enregistrer en un fichier ASCII le manuel original, puis le mettre au propre.

Sur ce premier matériau ainsi obtenu, faites agir **FRENCH ASSISTANT**. Phrase par phrase, la traduction s'affichera, vous proposant des variantes et l'amélioration de la version brute. La production de ce second matériau est ordinairement longue et fastidieuse : l'étape que vous venez de franchir a coûté trois fois moins de temps et vous laisse, en conséquence, le loisir d'exercer vos talents.

Au moyen de votre traitement de textes favori (WordCraft, WordPerfect, Word pour Windows, etc...), peaufinez les tournures, stylisez les phrases, sculptez les paragraphes : l'ouvrage touche à sa fin.

Il reste à le relire : confiez ce labeur à **HUGO PLUS** ou à **GRAMR**, ou les deux, pour qu'aucune faute ne subsiste.

C'est alors que vous êtes prêt pour l'impression, à condition toutefois de procéder à la mise en page et à l'illustration de votre manuel. C'est là que vous ferez intervenir **GEM DESKTOP PUBLISHER, WORD** pour Windows, **WORCRAFT** Elite ou **ARTLINE 2**, selon vos goûts et votre environnement informatique.

Pour imprimer, il vous reste une alternative : produire sur votre imprimante laser ou transmettre à votre imprimeur. Comme par hasard, son système informatique (Macintosh) est incompatible avec le vôtre (PC). Grâce au système **OPTION BOARD DELUXE**, créez une disquette Macintosh sur votre PC et remettez-la à votre imprimeur.

Et voilà !

Nous conseillons, fournissons et supportons les maillons de cette chaîne d'outils productifs, à l'exception des instruments (ordinateur, scanner, imprimante). □

FINESSE

1 ***Vous avez installé DR DOS 5.0, votre disquette B: est au format 3,5" et vous constatez des bizarreries.***

DR-DOS est averti d'un éventuel échange de disquette par la platine. Certaines platines ne gèrent pas cette "mise à jour", et c'est le cas que vous avez rencontré. Pour pallier cette difficulté, DR-DOS procure le moyen d'être prévenu que telle platine ne le renseignera pas sur ce qu'un changement de disquette est intervenu. A cette fin, vous devez ajouter dans votre fichier CONFIG.SYS l'instruction suivante :

DEVICE = C :/DRDOS/DRIVER.SYS/D:1/F:2/S:9/T:80
(*cf. MANUEL DE L'UTILISATEUR, pp. 334-335*)

où l'on suppose que :

DRIVER. SYS est dans le répertoire DRDOS
que votre disquette 3,5" est B: (option /D:1)
que son format est 720 kO (option /F:2)
qu'elle contient 9 secteurs par piste (option /S:9)
et 80 pistes (option /T:80)

le fait de ne pas mentionner l'option /C alerte DR-DOS 5.0 sur ce que votre platine ne le renseigne pas sur les changements de disquette.

D'autre part, il se peut que le problème soit lié à certains BIOS. Dans ce cas, une solution existe et consiste à utiliser l'instruction DRIVPARM dans le fichier CONFIG.SYS (cf. Manuel de l'utilisateur, pp. 307-308). Cette instruction permet de redéfinir les paramètres d'un lecteur de disquettes connu du système. C'est le cas qui s'applique aux machines de type AT3, leur configuration étant enregistrée dans une mémoire entretenue. Pour résoudre ce problème, remplacez dans CONFIG.SYS l'instruction:

DEVICE = C:/DRDOS/DRIVER.SYS/D:1/F:2/S:9/T:80
par l'instruction : DRIVPARM /D:1/F:2/S:9/T:80

2 ***Le logiciel anti-virus SCAN 6.8V74 de McAfee Associates a détecté le virus Swedish Stone dans le secteur de démarrage de votre disquette originale DR DOS 5.0***

N'ayez crainte : c'est une erreur du logiciel SCAN. Elle a été corrigée sitôt cette découverte faite. La version corrigée date du 15-2-91 et s'appelle SCAN 6.8B74. Merci à notre client de Pau de nous avoir alertés. □

COUP D'ŒIL

Maintenance des supports magnétiques et des fichiers

NORTON UTILITIES v. 5.0

Nous avons testé pour vous cette nouvelle version comportant de nouveaux outils et des performances accrues. NORTON UTILITIES 5.0 sait maintenant prolonger la durée de vie de vos disques durs, éviter les effacements accidentels de fichiers en les protégeant (idéal contre les virus), et récupérer des fichiers endommagés dans leur formats natifs (DBase ou Lotus 1-2-3).

Certains outils de la version antérieure ont été améliorés. Disk Doctor (véritable boite à outils à lui seul) est plus sûr et beaucoup plus performant; Speed Disk (Défragmentation de fichiers) est remarquable de rapidité et d'efficacité. NORTON UTILITIES 5.0 est un ensemble complet d'utilitaires disque fiables supportant désormais la souris.

BECKER TOOLS pour Windows

Ergonomie

• Becker Tools possède trois niveaux d'utilisation (débutant, intermédiaire et expert).

• Deux fenêtres de travail : source et cible.
• Sélection et traitement des fichiers/ répertoires par groupe.
• Accès direct aux applications et accessoires Windows dans un menu utilisateur.

Opérations sur fichiers et répertoires

• copie, effacement, déplacement des fichiers.
• Comparaison des données.
• Modification des attributs (par groupe ou individuellement).
• Compactage et décompactage des fichiers et répertoires avec un gain moyen de 70 % et protection par mot de passe.
• Recherche et extraction de fichiers par filtre ou par groupe.
• Récupération des données intactes au sein d'un fichier endommagé.

Opérations sur disques

• Archivage de fichiers et répertoires sélectionnés par blocs de tailles paramétrable (360ko, 1,2MO, 1,44MO...).
• Dépannage : contrôle physique du disque.
• Accès et modification de données secteur par secteur. Chargement et sauvegarde du secteur d'amorçage, de la FAT, du répertoire racine...
-•Purge des fichiers effacés sur disques et disquettes.

Autres outils

• Editeur de texte : lecture et édition des fichiers ASCII, ANSI, et Hexa. Couper/Coller, Rechercher/Remplacer.

• Verrouillage du clavier par mot de passe. □

TARIF AU 01.04.91

Logiciels	*Langue (*)*	*Prix HT Public*	*Prix HT Logone*
CopyWrite Septembre 1990	Doc		895,00
Mise à jour Copywrite	Doc		395,00
Disk Explorer	Doc		895,00
Analyzer	Doc		1695,00
Antidote	Doc		745,00
PC Tools Deluxe 6	Fr	1690,00	1350,00
Option Board Deluxe 5.4	Doc		1595,00
Câble Option board pour PS/2 (25 et 30)			250,00
Norton Utilities 5.0	VO		1295,00
Norton Utilities 5.0	Fr	1750,00	1750,00
Norton Commander 3.0	Fr	990,00	990,00
Câble Norton Commander (liaison inter-oridinateurs)		200,00	200,00
Norton Backup 1.0	Fr	1750,00	1750,00
Norton Backup 1.1	VO		995,00
Hugo + (Correcteur grammatical)		990,00	750,00
WordCraft Elite (terxte + P.A.O)	Fr	4995,00	3995,00
DR DOS 5.0	VO	990,00	990,00
DR DOS 5.0	Fr	990,00	990,00
Concurrent DOS 3.0 (multitâche, Multiposte)	Fr	4740,00	4740,00
Multi User DOS 5.0 (multitâche, Multipose)	Fr	5800,00	5800,00
Artline 2 (Création de documents graphiques)	Fr	4950,00	4950,00
GEM Desktop Publisher (P.A.O.)	Fr	2980,00	2980,00
GEM Scan (gestion scanner)	Fr	950,00	950,00
101 Jeux	Fr	501,68	501,68
101 Windows	Fr	501,68	501,68
Overcom	Fr	790,00	790,00
Memotel II	Fr	2490,00	2490,00
Hometel	Fr	1950,00	1950,00
Gram'R (Correcteur grammatical)	Fr	1490,00	1237,00
FontMaker	Fr	950,00	950,00
Becker Tools Windows	Fr	501,69	501,69
Pack Antivirus Window	Fr	333,05	333,05
Lasertwin	Fr	1790,00	1790,00
Superfonts 25/1	Fr	2690,00	2690,00
Windows 3.0	Fr	1990,00	1990,00
Word Windows	Fr	4990,00	4990,00
Excel Windows	Fr	4990,00	4990,00
French Assistant	VO	990,00	795,00
Omnipage pour Windows v. 3.0 (micro OCR)	VO	8995,00	6395,00
QA Plus	VO	1690,00	1395,00

Service "Fouineur" : Pour tout autre logiciel, nous contacter (Borland, Corel, Funk Software, Lotus, Ontrack, Quaterdeck, Travelling Software, WordCarft International, etc...).

(*)
Doc : Logiciel livré en version originale accompagné d'une traduction française du manuel
Fr : Logiciel et manuel en français
VO : Logiciel et manuel en version originale (anglais)

```
DLGTEMPLATE IDM_DIREND LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    DIALOG "", IDM_DIREND, 149, 18, 101, 27, FS_NOBYTEALIGN | FS_DLGBORDER |
              WS_VISIBLE | WS_SAVEBITS
    BEGIN
        CONTROL "Annul ", 256, 30, 2, 38, 12, WC_BUTTON, BS_PUSHBUTTON |
                BS_DEFAULT | WS_TABSTOP | WS_VISIBLE
        CONTROL "Directory Complet ", 257, 9, 16, 84, 8, WC_STATIC, SS_TEXT |
                DT_LEFT | DT_TOP | WS_GROUP | WS_VISIBLE
    END
END
```

[LISTING DIX-NEUF]

```
HEAPSIZE 16384
STACKSIZE 16384
EXPORTS
    WindowProc
    ChildWindowProc


[FILE PCKERMIT]

OS2DEF.SYM: OS2DEF.DEF
    M2 OS2DEF.DEF/OUT:OS2DEF.SYM
OS2DEF.OBJ: OS2DEF.MOD OS2DEF.SYM
    M2 OS2DEF.MOD/OUT:OS2DEF.OBJ
PMWIN.SYM: PMWIN.DEF OS2DEF.SYM
    M2 PMWIN.DEF/OUT:PMWIN.SYM
PMWIN.OBJ: PMWIN.MOD OS2DEF.SYM PMWIN.SYM
    M2 PMWIN.MOD/OUT:PMWIN.OBJ
KH.SYM: KH.DEF
    M2 KH.DEF/OUT:KH.SYM
KH.OBJ: KH.MOD KH.SYM
    M2 KH.MOD/OUT:KH.OBJ
SHELL.SYM: SHELL.DEF PMWIN.SYM OS2DEF.SYM
    M2 SHELL.DEF/OUT:SHELL.SYM
TERM.SYM: TERM.DEF
    M2 TERM.DEF/OUT:TERM.SYM
PAD.SYM: PAD.DEF PMWIN.SYM
    M2 PAD.DEF/OUT:PAD.SYM
DATALINK.SYM: DATALINK.DEF PAD.SYM PMWIN.SYM
    M2 DATALINK.DEF/OUT:DATALINK.SYM
PMAVIO.SYM: PMAVIO.DEF PMWIN.SYM OS2DEF.SYM
    M2 PMAVIO.DEF/OUT:PMAVIO.SYM
PMAVIO.OBJ: PMAVIO.MOD PMAVIO.SYM
    M2 PMAVIO.MOD/OUT:PMAVIO.OBJ
PMGPI.SYM: PMGPI.DEF OS2DEF.SYM
    M2 PMGPI.DEF/OUT:PMGPI.SYM
PMGPI.OBJ: PMGPI.MOD OS2DEF.SYM PMGPI.SYM
    M2 PMGPI.MOD/OUT:PMGPI.OBJ
COMMPORT.SYM: COMMPORT.DEF
    M2 COMMPORT.DEF/OUT:COMMPORT.SYM
COMMPORT.OBJ: COMMPORT.MOD COMMPORT.SYM
    M2 COMMPORT.MOD/OUT:COMMPORT.OBJ
FILES.SYM: FILES.DEF
    M2 FILES.DEF/OUT:FILES.SYM
PCKERMIT.OBJ: PCKERMIT.MOD SHELL.SYM KH.SYM PMWIN.SYM OS2DEF.SYM
    M2 PCKERMIT.MOD/OUT:PCKERMIT.OBJ
SCREEN.SYM: SCREEN.DEF PMAVIO.SYM
    M2 SCREEN.DEF/OUT:SCREEN.SYM
SCREEN.OBJ: SCREEN.MOD SCREEN.SYM
    M2 SCREEN.MOD/OUT:SCREEN.OBJ
FILES.OBJ: FILES.MOD FILES.SYM
    M2 FILES.MOD/OUT:FILES.OBJ
SHELL.OBJ: SHELL.MOD COMMPORT.SYM KH.SYM PMGPI.SYM PMWIN.SYM PMAVIO.SYM -
SCREEN.SYM DATALINK.SYM PAD.SYM TERM.SYM OS2DEF.SYM SHELL.SYM
    M2 SHELL.MOD/OUT:SHELL.OBJ
TERM.OBJ:  TERM.MOD  COMMPORT.SYM  KH.SYM  SHELL.SYM  PMWIN.SYM  SCREEN.SYM
TERM.SYM
    M2 TERM.MOD/OUT:TERM.OBJ
PAD.OBJ: PAD.MOD DATALINK.SYM KH.SYM SHELL.SYM PMWIN.SYM COMMPORT.SYM -
FILES.SYM OS2DEF.SYM SCREEN.SYM PAD.SYM
    M2 PAD.MOD/OUT:PAD.OBJ
DATALINK.OBJ: DATALINK.MOD KH.SYM PAD.SYM COMMPORT.SYM SHELL.SYM PMWIN.SYM -
OS2DEF.SYM SCREEN.SYM DATALINK.SYM
    M2 DATALINK.MOD/OUT:DATALINK.OBJ
PCKERMIT.res: PCKERMIT.rc PCKERMIT.h PCKERMIT.ico
    rc -r PCKERMIT.rc
PCKERMIT.EXE: OS2DEF.OBJ PMWIN.OBJ KH.OBJ PMAVIO.OBJ PMGPI.OBJ COMMPORT.OBJ -
PCKERMIT.OBJ SCREEN.OBJ FILES.OBJ SHELL.OBJ TERM.OBJ PAD.OBJ DATALINK.OBJ
    LINK @PCKERMIT.LNK
    rc PCKERMIT.res
PCKERMIT.exe: PCKERMIT.res
    rc PCKERMIT.res



[FICHIER PCKERMIT.LNK]

KH.OBJ+
pckermit.OBJ+
SCREEN.OBJ+
COMMPORT.OBJ+
FILES.OBJ+
SHELL.OBJ+
TERM.OBJ+
PAD.OBJ+
DATALINK.OBJ

pckermit
pckermit
PM+
M2LIB+
DOSCALLS+
OS2
pckermit.edf


[FICHIER PAD.MOD]

IMPLEMENTATION MODULE PAD;   (* Assembleur/désassembleur de paquets *)

   FROM SYSTEM IMPORT
      ADR;

   FROM Storage IMPORT
      ALLOCATE, DEALLOCATE;

   FROM Screen IMPORT
      ClrScr, WriteString, WriteInt, WriteHex, WriteLn;

   FROM OS2DEF IMPORT
      HIWORD, LOWORD;

   FROM DosCalls IMPORT
      ExitType, DosExit;

   FROM Strings IMPORT
      Length, Assign;

   FROM FileSystem IMPORT
      File;

   FROM Directories IMPORT
      FileAttributes, AttributeSet, DirectoryEntry, FindFirst, FindNext;

   FROM Files IMPORT
      Status, FileType, Open, Create, CloseFile, Get, Put, DoWrite;

   FROM PMWIN IMPORT
      MPARAM, MPFROM2SHORT, WinPostMsg;

   FROM Shell IMPORT
      ChildFrameWindow, comport;

   FROM KH IMPORT
      COM_OFF;

   FROM DataLink IMPORT
      FlushUART, SendPacket, ReceivePacket;

   FROM SYSTEM IMPORT
      BYTE;

   IMPORT ASCII;


   CONST
      myMAXL = 94;
      myTIME = 10;
      myNPAD = 0;
      myPADC = 0C;
      myEOL  = 0C;
      myQCTL = '#';
      myQBIN = '&';
      myCHKT = '1';      (* checksum un caractère *)
      MAXtrys = 5;
      (* du "DEFINITION MODULE":
      PAD_Quit = 0;  *)
      PAD_SendPacket = 1;
      PAD_ResendPacket = 2;
      PAD_NoSuchFile = 3;
      PAD_ExcessiveErrors = 4;
      PAD_ProbClSrcFile = 5;
      PAD_ReceivedPacket = 6;
      PAD_Filename = 7;
      PAD_RequestRepeat = 8;
      PAD_DuplicatePacket = 9;
      PAD_UnableToOpen = 10;
      PAD_ProbClDestFile = 11;
      PAD_ErrWrtFile = 12;
      PAD_Msg = 13;


   TYPE
      (* du Module Definition:
      PacketType = ARRAY [1..100] OF CHAR;
      *)
      SMALLSET = SET OF [0..7];   (* un octet *)


   VAR
      yourMAXL : INTEGER;   (* longueur maxi d'un paquet - jusqu'à 94 *)
      yourTIME : INTEGER;   (* time out - en secondes *)
      (* Du module Definition
      yourNPAD : INTEGER;   (* nb de caractères de remplissage *)
      yourPADC : CHAR;   (* caractère de remplissage *)
      yourEOL  : CHAR;   (* End Of Line - fin de ligne *)
      *)
      yourQCTL : CHAR;   (* caractère d'annotation des contrôles '#' *)
      yourQBIN : CHAR;   (* caractère d'annotation binaire '&' *)
      yourCHKT : CHAR;   (* type de vérification : 1 = checksum, etc. *)
      sF, rF : File;   (* fichiers envoyés/reçus *)

      InputFileOpen : BOOLEAN;
      rFname : ARRAY [0..20] OF CHAR;
```

```
    sP, rP : PacketType;   (* paquets envoyés/reçus *)
    sSeq, rSeq : INTEGER;   (* sequence *)
    PktNbr : INTEGER;   (* numéro du paquet en cours -
                           pas de répétition jusqu'à 32,000 *)
    ErrorMsg : ARRAY [0..40] OF CHAR;


PROCEDURE PtrToStr (mp : MPARAM; VAR s : ARRAY OF CHAR);
(* Converti un pointeur sur une chaine en chaine *)

   TYPE
      PC = POINTER TO CHAR;

   VAR
      p : PC;
      i : CARDINAL;
      c : CHAR;

   BEGIN
      i := 0;
      REPEAT
         p := PC (mp);
         c := p^;
         s[i] := c;
         INC (i);
         INC (mp);
      UNTIL c = 0C;
   END PtrToStr;


PROCEDURE DoPADMsg (mp1, mp2 : MPARAM);
(* Affichage des messages du module PAD *)

   VAR
      Message : ARRAY [0..40] OF CHAR;

   BEGIN
      CASE LOWORD (mp1) OF
         PAD_SendPacket:
            WriteString ("Envoie paquet ");
            WriteInt (LOWORD (mp2), 5);
            WriteString ("  (ID: ");   WriteHex (HIWORD (mp2), 2);
            WriteString ("h)");
      | PAD_ResendPacket:
            WriteString ("ERREUR - Re-envoie:");   WriteLn;
            WriteString ("      Packet #");
            WriteInt (LOWORD (mp2), 5);
            WriteString ("  (ID: ");   WriteHex (HIWORD (mp2), 2);
            WriteString ("h)");
      | PAD_NoSuchFile:
            WriteString ("Fichier inconnu: ");
            PtrToStr (mp2, Message);   WriteString (Message);
      | PAD_ExcessiveErrors:
            WriteString ("Trop d'erreurs ...");
      | PAD_ProbClSrcFile:
            WriteString ("Erreur à la fermeture du fichier source ...");
      | PAD_ReceivedPacket:
            WriteString ("Reçoit Paquet ");
            WriteInt (LOWORD (mp2), 5);
            WriteString ("  (ID: ");   WriteHex (HIWORD (mp2), 2);
            WriteString ("h)");
      | PAD_Filename:
            WriteString ("Fichier = ");
            PtrToStr (mp2, Message);   WriteString (Message);
      | PAD_RequestRepeat:
            WriteString ("ERREUR - Demande re-émission:");   WriteLn;
            WriteString ("       Paquet #");
            WriteInt (LOWORD (mp2), 5);
            WriteString ("  (ID: ");   WriteHex (HIWORD (mp2), 2);
            WriteString ("h)");
      | PAD_DuplicatePacket:
            WriteString ("Ignore paquet dupliqué:");   WriteLn;
            WriteString ("         Paquet ");
            WriteString ("  (ID: ");   WriteHex (HIWORD (mp2), 2);
            WriteString ("h)");
      | PAD_UnableToOpen:
            WriteString ("Impossible ouvrir fichier: ");
            PtrToStr (mp2, Message);   WriteString (Message);
      | PAD_ProbClDestFile:
            WriteString ("Erreur fermeture fichier : ");
            PtrToStr (mp2, Message);   WriteString (Message);
      | PAD_ErrWrtFile:
            WriteString ("Erreur écriture dans fichier : ");
            PtrToStr (mp2, Message);   WriteString (Message);
      | PAD_Msg:
            PtrToStr (mp2, Message);   WriteString (Message);
      ELSE
         (* Ne fait rien *)
      END;
      WriteLn;
   END DoPADMsg;


PROCEDURE CloseInput;
(* Ferme le fichier en lecture s'il existe. Réinitialise le flag associé *)
   BEGIN
      IF InputFileOpen THEN
         IF CloseFile (sF, Input) = Done THEN
            InputFileOpen := FALSE;
         ELSE
            WinPostMsg (ChildFrameWindow, WM_PAD,
               MPFROM2SHORT (PAD_ProbClSrcFile, 0),

                  ADR (sFname));
         END;
      END;
   END CloseInput;


PROCEDURE NormalQuit;
(* Quitte une tâche, envoie un message à la fenêtre *)
   BEGIN
      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_Quit, 0), 0);
      DosExit (EXIT_THREAD, 0);
   END NormalQuit;


PROCEDURE ErrorQuit;
(* Quitte une tâche, envoie un message à la fenêtre *)
   BEGIN
      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_Error, 0), 0);
      DosExit (EXIT_THREAD, 0);
   END ErrorQuit;


PROCEDURE ByteXor (a, b : BYTE) : BYTE;
   BEGIN
      RETURN BYTE (SMALLSET (a) / SMALLSET (b));
   END ByteXor;


PROCEDURE Char (c : INTEGER) : CHAR;
(* convertit un nombre 0-94 en caractère imprimable *)
   BEGIN
      RETURN (CHR (CARDINAL (ABS (c) + 32)));
   END Char;


PROCEDURE UnChar (c : CHAR) : INTEGER;
(* convertit un caractère en nombre *)
   BEGIN
      RETURN (ABS (INTEGER (ORD (c)) - 32));
   END UnChar;


PROCEDURE TellError (Seq : INTEGER);
(* Envoie un paquet d'erreur *)
   BEGIN
      sP[1] := Char (15);
      sP[2] := Char (Seq);
      sP[3] := 'E';   (* Paquet de type E *)
      sP[4] := 'R';   (* Début du message d'erreur *)
      sP[5] := 'e';
      sP[6] := 'm';
      sP[7] := 'o';
      sP[8] := 't';
      sP[9] := 'e';
      sP[10] := ' ';
      sP[11] := 'A';
      sP[12] := 'b';
      sP[13] := 'o';
      sP[14] := 'r';
      sP[15] := 't';
      sP[16] := 0C;
      SendPacket (sP);
   END TellError;


PROCEDURE ShowError (p : PacketType);
(* Affiche le contenu d'un paquet d'erreur *)

   VAR
      i : INTEGER;

   BEGIN
      FOR i := 4 TO UnChar (p[1]) DO
         ErrorMsg[i - 4] := p[i];
      END;
      ErrorMsg[i - 4] := 0C;
      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_Msg, 0), ADR (ErrorMsg));
   END ShowError;


PROCEDURE youInit (type : CHAR);
(* JE VOUS initialise en émission et réception *)
   BEGIN
      sP[1] := Char (11);   (* Lengueur *)
      sP[2] := Char (0);   (* Séquence *)
      sP[3] := type;
      sP[4] := Char (myMAXL);
      sP[5] := Char (myTIME);
      sP[6] := Char (myNPAD);
      sP[7] := CHAR (ByteXor (myPADC, 100C));
      sP[8] := Char (ORD (myEOL));
      sP[9] := myQCTL;
      sP[10] := myQBIN;
      sP[11] := myCHKT;
      sP[12] := 0C;   (* terminaison *)
      SendPacket (sP);
   END youInit;


PROCEDURE myInit;
(* VOUS M'initialisez en émission et réception *)

   VAR
      len : INTEGER;
```

ABONNEMENT *1 AN DISQUETTES LISTING*

❒ Je m'abonne pour **12** disquettes listing : **399 F** au lieu de **444 F**

Je règle par : ❒ chèque à l'ordre de **Debug Magazine**

❒ carte bancaire

Nom de la carte : .....

N° ..../..../..../..../..../..../..../..../..../..../..../..../..../..../ Date d'expiration /...../...../...../...../

Nom :..... Prénom : .....

Société :.....

Rue :.....

Lieu :.....

Code postal :..... ..... Ville : .....

Pays :..... Tél : .....

Date :..... Signature : .....

A renvoyer à **DEBUG MAGAZINE** - 3, rue de l'Atlas - 75019 Paris

COMMANDE *DISQUETTE LISTING N° 11*

❒ Je désire recevoir la disquette listing **DEBUG 11** au prix de **47 F** pièce

Je règle par : ❒ chèque à l'ordre de **Debug Magazine**

❒ contre remboursement (ajouter **40 F** de frais)

❒ carte bancaire

Nom de la carte : .....

N° ..../..../..../..../..../..../..../..../..../..../..../..../..../..../ Date d'expiration /...../...../...../...../

Nom :..... Prénom : .....

Société :.....

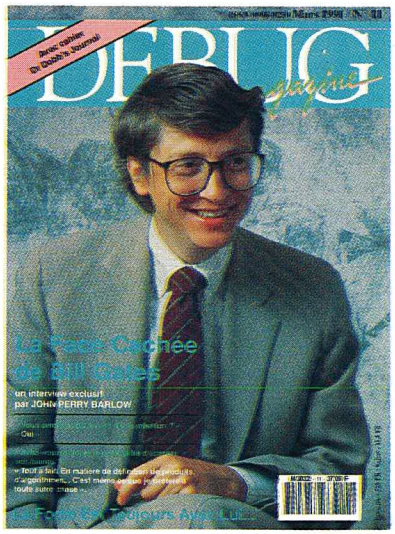
Rue :.....

Lieu :.....

Code postal :..... ..... Ville : .....

Pays :..... Tél : .....

Date :..... Signature : .....



Anciennes disquettes à **47 F:** ❒ n° 1 ❒ n° 2 ❒ n° 3 ❒ n° 4 ❒ n° 5 ❒ n° 6 ❒ n° 7 ❒ n° 8 ❒ n° 9 ❒ n° 10

❒ SCRIBE (150 F) ❒ SDIR (45 F)

A renvoyer à **DEBUG MAGAZINE** - 3, rue de l'Atlas - 75019 Paris

SERVICE LECTEURS

DEBUG Magazine n° 11

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Nom : ..... Prénom : .....
Société : .....
Rue : .....
Lieu : .....
Code postal : ..... Ville : .....
Pays : ..... Téléphone : .....

A REMPLIR OBLIGATOIREMENT SI VOUS AVEZ DEMANDÉ UNE DOC.

Le Service Lecteurs de DEBUG Magazine transmettra aux intéressés les demandes d'envoi de documentation dont vous aurez coché les cases.

Notre service vous permet également de donner votre opinion sur les articles parus. Cochez les cases correspondant à votre jugement sur l'article concerné (les petits numéros au début de chaque article, pas les numéros des pages) ; cela nous permettra d'améliorer la qualité de la revue. Merci !

pliez ici en second et collez l'autre côté au ruban adhésif.

AFFRANCHIR
AU TARIF
LETTRE..

DEBUG MAGAZINE
3, rue de l'Atlas
75019 Paris

pliez ici en premier.

QUESTIONNAIRE – LECTEUR

Année de naissance : 19__
Sexe : ☐ M ☐ F

Votre secteur d'activité :

- ☐ Agriculture
- ☐ Enseignement
- ☐ Autres Fonction Publique, Collectivité locale
- ☐ Association
- ☐ Energie
- ☐ Métallurgie
- ☐ Chimie, verre, plastique
- ☐ Construction automobile, mécanique, navale, aéronautique
- ☐ Electronique
- ☐ Bâtiment, travaux publics
- ☐ Autres secteurs industriels
- ☐ Commerce
- ☐ Transports
- ☐ Télécommunications
- ☐ SSII
- ☐ Autres services aux entreprises
- ☐ Hôtellerie, réparation et autres services aux particuliers
- ☐ Banque, Assurance
- ☐ Autres secteurs commerciaux
- ☐ Informatique

Nombre de salariés dans l'établissement : ____

Votre profession :

- ☐ Propriétaire
- ☐ PDG-DG-Gérant
- ☐ Cadre
- ☐ Employé
- ☐ Ouvrier
- ☐ Retraité
- ☐ Autre
- ☐ Etudiant

Votre matériel :

- ☐ IBM ou compatible
- ☐ Apple, MAC
- ☐ Mini
- ☐ Mainframe
- ☐ Capacité de votre disque dur __ Mo
- ☐ Type de l'écran _____
- ☐ Valeur totale de votre installation _____ F TTC

L'utilisation de votre matériel :

- ☐ privée
- ☐ professionnelle
- ☐ les deux

Etes-vous satisfait de votre équipement actuel ? ☐ Oui ☐ Non
Allez-vous l'étendre ? ☐ Oui ☐ Non

Vos commentaires :

```
   BEGIN
      len := UnChar (rP[1]);
      IF len >= 4 THEN
         yourMAXL := UnChar (rP[4]);
      ELSE
         yourMAXL := 94;
      END;
      IF len >= 5 THEN
         yourTIME := UnChar (rP[5]);
      ELSE
         yourTIME := 10;
      END;
      IF len >= 6 THEN
         yourNPAD := UnChar (rP[6]);
      ELSE
         yourNPAD := 0;
      END;
      IF len >= 7 THEN
         yourPADC := CHAR (ByteXor (rP[7], 100C));
      ELSE
         yourPADC := 0C;
      END;
      IF len >= 8 THEN
         yourEOL := CHR (UnChar (rP[8]));
      ELSE
         yourEOL := 0C;
      END;
      IF len >= 9 THEN
         yourQCTL := rP[9];
      ELSE
         yourQCTL := 0C;
      END;
      IF len >= 10 THEN
         yourQBIN := rP[10];
      ELSE
         yourQBIN := 0C;
      END;
      IF len >= 11 THEN
         yourCHKT := rP[11];
         IF yourCHKT # myCHKT THEN
            yourCHKT := '1';
         END;
      ELSE
         yourCHKT := '1';
      END;
   END myInit;


PROCEDURE SendInit;
   BEGIN
      youInit ('S');
   END SendInit;


PROCEDURE SendFileName;

   VAR
      i, j : INTEGER;

   BEGIN
      (* envoie nom du fichier *)
      i := 4;    j := 0;
      WHILE sFname[j] # 0C DO
         sP[i] := sFname[j];
         INC (i);    INC (j);
      END;
      sP[1] := Char (j + 3);
      sP[2] := Char (sSeq);
      sP[3] := 'F';    (* paquet nom de fichier *)
      sP[i] := 0C;
      SendPacket (sP);
   END SendFileName;


PROCEDURE SendEOF;
   BEGIN
      sP[1] := Char (3);
      sP[2] := Char (sSeq);
      sP[3] := 'Z';    (* fin de fichier *)
      sP[4] := 0C;
      SendPacket (sP);
   END SendEOF;


PROCEDURE SendEOT;
   BEGIN
      sP[1] := Char (3);
      sP[2] := Char (sSeq);
      sP[3] := 'B';    (* break - fin de transmission *)
      sP[4] := 0C;
      SendPacket (sP);
   END SendEOT;


PROCEDURE GetAck() : BOOLEAN;
(* Attend un Ack - Réessaie si NACK ou timeout *)

   VAR
      Type : CHAR;
      Seq : INTEGER;
      retrys : INTEGER;

       AckOK : BOOLEAN;

      BEGIN
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_SendPacket, 0),
            MPFROM2SHORT (PktNbr, sSeq));

         retrys := MAXtrys;
         LOOP
            IF Aborted THEN
               TellError (sSeq);
               CloseInput;
               ErrorQuit;
            END;
            IF ReceivePacket (rP) THEN
               Seq := UnChar (rP[2]);
               Type := rP[3];
               IF (Seq = sSeq) AND (Type = 'Y') THEN
                  AckOK := TRUE;
               ELSIF (Seq = (sSeq + 1) MOD 64) AND (Type = 'N') THEN
                  AckOK := TRUE;    (* NAK for (n + 1) taken as ACK for n *)
               ELSIF Type = 'E' THEN
                  ShowError (rP);
                  AckOK := FALSE;
                  retrys := 0;
               ELSE
                  AckOK := FALSE;
               END;
            ELSE
               AckOK := FALSE;
            END;
            IF AckOK OR (retrys = 0) THEN
               EXIT;
            ELSE
               WinPostMsg (ChildFrameWindow, WM_PAD,
                  MPFROM2SHORT (PAD_ResendPacket, 0),
                  MPFROM2SHORT (PktNbr, sSeq));

               DEC (retrys);
               FlushUART;
               SendPacket (sP);
            END;
         END;

         IF AckOK THEN
            INC (PktNbr);
            sSeq := (sSeq + 1) MOD 64;
            RETURN TRUE;
         ELSE
            RETURN FALSE;
         END;
      END GetAck;


   PROCEDURE GetInitAck() : BOOLEAN;
   (* configuration de la station distante *)
      BEGIN
         IF GetAck() THEN
            myInit;
            RETURN TRUE;
         ELSE
            RETURN FALSE;
         END;
      END GetInitAck;


   PROCEDURE Send;
   (* Envoie un ou plusiers fichiers: sFname peut être ambigu *)

      TYPE
         LP = POINTER TO LIST;    (* liste de fichiers *)
         LIST = RECORD
                   fn : ARRAY [0..20] OF CHAR;
                   next : LP;
                END;

      VAR
         gotFN : BOOLEAN;
         attr : AttributeSet;
         ent : DirectoryEntry;
         front, back, t : LP;    (* ajoute à la fin de la queue,
                                    supprime au début *)

      BEGIN
         Aborted := FALSE;
         InputFileOpen := FALSE;

         front := NIL;    back := NIL;
         attr := AttributeSet {};    (* fichiers normaux seulement *)
         IF Length (sFname) = 0 THEN
            WinPostMsg (ChildFrameWindow, WM_PAD,
               MPFROM2SHORT (PAD_Msg, 0),
               ADR ("Pas de fichier..."));
            ErrorQuit;
         ELSE
            gotFN := FindFirst (sFname, attr, ent);
            WHILE gotFN DO    (* Construit une liste de noms de fichiers *)
               ALLOCATE (t, SIZE (LIST));
               Assign (ent.name, t^.fn);
               t^.next := NIL;
               IF front = NIL THEN
                  front := t;    (* commence au début de la queue vide *)
               ELSE
                  back^.next := t;    (*  sinon à la fin *)

               END;
               back := t;
```

```
            gotFN := FindNext (ent);
         END;
      END;

      IF front = NIL THEN
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_NoSuchFile, 0),
            ADR (sFname));
         ErrorQuit;
      ELSE
         sSeq := 0;   PktNbr := 0;
         FlushUART;
         SendInit;   (* ma configuration initiale *)
         IF NOT GetInitAck() THEN  (* lit votre configuration initiale *)
            WinPostMsg (ChildFrameWindow, WM_PAD,
               MPFROM2SHORT (PAD_ExcessiveErrors, 0),
               MPFROM2SHORT (0, 0));
            ErrorQuit;
         END;

         WHILE front # NIL DO   (* envoie les fichiers *)
            Assign (front^.fn, sFname);
            PktNbr := 1;
            Send1;
            t := front;
            front := front^.next;
            DEALLOCATE (t, SIZE (LIST));
         END;
      END;

      SendEOT;
      IF NOT GetAck() THEN
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_ExcessiveErrors, 0),
            MPFROM2SHORT (0, 0));
         CloseInput;
         ErrorQuit;
      END;
      NormalQuit;
   END Send;


PROCEDURE Send1;
(* Envoie un fichier: sFname *)

   VAR
      ch : CHAR;
      i : INTEGER;

   BEGIN
      IF Open (sF, sFname) = Done THEN
         InputFileOpen := TRUE;
      ELSE;
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_NoSuchFile, 0),
            ADR (sFname));
         ErrorQuit;
      END;

      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_Filename, 0),
         ADR (sFname));
      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_Msg, 0),
         ADR ("(<ESC> pour interrompre le transfert.)"));

      SendFileName;
      IF NOT GetAck() THEN
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_ExcessiveErrors, 0),
            MPFROM2SHORT (0, 0));
         CloseInput;
         ErrorQuit;
      END;

      (* envoie fichier *)
      i := 4;
      LOOP
         IF Get (sF, ch) = EOF THEN (* envoie paquet courant et termine *)
            sP[1] := Char (i - 1);
            sP[2] := Char (sSeq);
            sP[3] := 'D';   (* paquet de données *)
            sP[i] := 0C;   (* fin de paquet *)
            SendPacket (sP);
            IF NOT GetAck() THEN
               WinPostMsg (ChildFrameWindow, WM_PAD,
                  MPFROM2SHORT (PAD_ExcessiveErrors, 0),
                  MPFROM2SHORT (0, 0));
               CloseInput;
               ErrorQuit;
            END;
            SendEOF;
            IF NOT GetAck() THEN
               WinPostMsg (ChildFrameWindow, WM_PAD,
                  MPFROM2SHORT (PAD_ExcessiveErrors, 0),
                  MPFROM2SHORT (0, 0));
               CloseInput;
               ErrorQuit;
            END;
            EXIT;
         END;

         IF i >= (yourMAXL - 4) THEN   (* envoie paquet courant *)
            sP[1] := Char (i - 1);
            sP[2] := Char (sSeq);
            sP[3] := 'D';
            sP[i] := 0C;
            SendPacket (sP);
            IF NOT GetAck() THEN
               WinPostMsg (ChildFrameWindow, WM_PAD,
                  MPFROM2SHORT (PAD_ExcessiveErrors, 0),
                  MPFROM2SHORT (0, 0));
               CloseInput;
               ErrorQuit;
            END;
            i := 4;
         END;
         (* ajoute un caractère au paquet, modifie compteur caractères *)
         IF ch > 177C THEN   (* doit être annoté (QBIN) *)
            (* inverse bit 7 *)
            ch := CHAR (ByteXor (ch, 200C));
            sP[i] := myQBIN;   INC (i);
         END;
         IF (ch < 40C) OR (ch = 177C) THEN   (* annote (QCTL) *)
            (* inverse bit 6 *)
            ch := CHAR (ByteXor (ch, 100C));
            sP[i] := myQCTL;   INC (i);
         END;
         IF (ch = myQCTL) OR (ch = myQBIN) THEN
            (* doit être envoyé annoté *)
            sP[i] := myQCTL;   INC (i);
         END;
         sP[i] := ch;   INC (i);
      END;   (* LOOP *)
      CloseInput;
   END Send1;

PROCEDURE ReceiveInit() : BOOLEAN;
(* je reçois mon initialisation de votre part *)

   VAR
      RecOK : BOOLEAN;
      trys : INTEGER;

   BEGIN
      trys := 1;
      LOOP
         IF Aborted THEN
            TellError (rSeq);
            ErrorQuit;
         END;
         RecOK := ReceivePacket (rP) AND (rP[3] = 'S');
         IF RecOK OR (trys = MAXtrys) THEN
            EXIT;
         ELSE
            INC (trys);
            SendNak;
         END;
      END;
      IF RecOK THEN
         myInit;
         RETURN TRUE;
      ELSE
         RETURN FALSE;
      END;
   END ReceiveInit;


PROCEDURE SendInitAck;
(* envoie un ACK suite à mon initialisation *)
(* que vous m'avez envoyée, et j'envoie la votre *)
   BEGIN
      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_ReceivedPacket, 0),
         MPFROM2SHORT (PktNbr, rSeq));
      INC (PktNbr);
      rSeq := (rSeq + 1) MOD 64;
      youInit ('Y');
   END SendInitAck;


PROCEDURE ValidFileChar (VAR ch : CHAR) : BOOLEAN;
(* teste si le caractère fait partie de 'A'..'Z', '0'..'9' *)
(* met les lettres en majuscule *)
   BEGIN
    ch := CAP (ch);
    RETURN ((ch >= 'A') AND (ch <= 'Z')) OR ((ch >= '0') AND (ch <= '9'));
   END ValidFileChar;

 TYPE
   HeaderType = (name, eot, fail);

PROCEDURE ReceiveHeader() : HeaderType;
(* reçoit un nom de fichier - le modifie pour qu'il respecte *)
(* l'environnement local si nécessaire *)

   VAR
      i, j, k : INTEGER;
      RecOK : BOOLEAN;
      trys : INTEGER;
   BEGIN
      trys := 1;
      LOOP
         IF Aborted THEN
            TellError (rSeq);
            ErrorQuit;
         END;
         RecOK := ReceivePacket (rP) AND ((rP[3] = 'F') OR (rP[3] = 'B'));
         IF trys = MAXtrys THEN
            RETURN fail;
```

```
      ELSIF RecOK AND (rP[3] = 'F') THEN
         i := 4;   (* début des données ici *)
         j := 0;   (* début du nom de fichier ici *)
         WHILE (ValidFileChar (rP[i])) AND (j < 8) DO
            rFname[j] := rP[i];
            INC (i);   INC (j);
         END;
         REPEAT
            INC (i);
         UNTIL (ValidFileChar (rP[i])) OR (rP[i] = 0C);
         rFname[j] := '.';   INC (j);
         k := 0;
         WHILE (ValidFileChar (rP[i])) AND (k < 3) DO
            rFname[j + k] := rP[i];
            INC (i);   INC (k);
         END;
         rFname[j + k] := 0C;
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_Filename, 0),
            ADR (rFname));
         RETURN name;
      ELSIF RecOK AND (rP[3] = 'B') THEN
         RETURN eot;
      ELSE
         INC (trys);
         SendNak;
      END;
   END;
END ReceiveHeader;


PROCEDURE SendNak;
   BEGIN
      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_RequestRepeat, 0),
         MPFROM2SHORT (PktNbr, rSeq));
      FlushUART;
      sP[1] := Char (3);   (* Longueur *)
      sP[2] := Char (rSeq);
      sP[3] := 'N';   (* ACK négatif *)
      sP[4] := 0C;
      SendPacket (sP);
   END SendNak;


PROCEDURE SendAck (Seq : INTEGER);
   BEGIN
      IF Seq # rSeq THEN
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_DuplicatePacket, 0),
            MPFROM2SHORT (0, rSeq));
      ELSE
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_ReceivedPacket, 0),
            MPFROM2SHORT (PktNbr, rSeq));
         rSeq := (rSeq + 1) MOD 64;
         INC (PktNbr);
      END;

      sP[1] := Char (3);
      sP[2] := Char (Seq);
      sP[3] := 'Y';   (* ACK positif *)
      sP[4] := 0C;
      SendPacket (sP);
   END SendAck;


PROCEDURE Receive;
(* Reçoit un ou plusieurs fichiers *)

   VAR
      ch, Type : CHAR;
      Seq : INTEGER;
      i : INTEGER;
      EOF, EOT, QBIN : BOOLEAN;
      trys : INTEGER;

   BEGIN
      Aborted := FALSE;

      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_Msg, 0),
         ADR ("Prêt à recevoir des fichiers..."));
      WinPostMsg (ChildFrameWindow, WM_PAD,
         MPFROM2SHORT (PAD_Msg, 0),
         ADR ("(<ESC> pour interrompre le transfert.)"));

      FlushUART;
      rSeq := 0;   PktNbr := 0;
      IF NOT ReceiveInit() THEN   (* votre configuration *)
         WinPostMsg (ChildFrameWindow, WM_PAD,
            MPFROM2SHORT (PAD_ExcessiveErrors, 0),
            MPFROM2SHORT (0, 0));
         ErrorQuit;
      END;

      SendInitAck;       (* j'envoie ma configuration *)
      EOT := FALSE;
      WHILE NOT EOT DO
         CASE ReceiveHeader() OF
            eot  : EOT := TRUE;   EOF := TRUE;
            | name : IF Create (rF, rFname) # Done THEN
                        WinPostMsg (ChildFrameWindow, WM_PAD,
                              MPFROM2SHORT (PAD_UnableToOpen, 0),
                              ADR (rFname));
                        ErrorQuit;
                     ELSE
                        PktNbr := 1;
                        EOF := FALSE;
                     END;
            | fail : WinPostMsg (ChildFrameWindow, WM_PAD,
                           MPFROM2SHORT (PAD_ExcessiveErrors, 0),
                           MPFROM2SHORT (0, 0));
                     ErrorQuit;
         END;
         SendAck (rSeq);   (* envoie un ACK pour le nom ou EOT *)
         trys := 1;   (* initialise *)
         WHILE NOT EOF DO
            IF Aborted THEN
               TellError (rSeq);
               ErrorQuit;
            END;
            IF ReceivePacket (rP) THEN
               Seq := UnChar (rP[2]);
               Type := rP[3];
               IF Type = 'Z' THEN
                  EOF := TRUE;
                  IF CloseFile (rF, Output) = Done THEN
                     (* fin de fichier normale *)
                  ELSE
                     WinPostMsg (ChildFrameWindow, WM_PAD,
                        MPFROM2SHORT (PAD_ProbClDestFile, 0),
                        ADR (rFname));
                     ErrorQuit;
                  END;
                  trys := 1;   (* bon paquuet - réinitialise *)
                  SendAck (rSeq);
               ELSIF Type = 'E' THEN
                  ShowError (rP);
                  ErrorQuit;
               ELSIF (Type = 'D') AND ((Seq + 1) MOD 64 = rSeq) THEN
               (* ignore paquet dupliqué, envoie un ACK *)
                  trys := 1;
                  SendAck (Seq);
               ELSIF (Type = 'D') AND (Seq = rSeq) THEN
                  (* recopie le paquet dans le buffer fichier *)
                  i := 4;   (* 1er octet de données *)
                  WHILE rP[i] # 0C DO
                     ch := rP[i];   INC (i);
                     IF ch = yourQBIN THEN
                        ch := rP[i];   INC (i);
                        QBIN := TRUE;
                     ELSE
                        QBIN := FALSE;
                     END;
                     IF ch = yourQCTL THEN
                        ch := rP[i];   INC (i);
                        IF (ch # yourQCTL) AND (ch # yourQBIN) THEN
                           ch := CHAR (ByteXor (ch, 100C));
                        END;
                     END;
                     IF QBIN THEN
                        ch := CHAR (ByteXor (ch, 200C));
                     END;
                     Put (ch);
                  END;

                  (* Ecrit fichier sur disque *)
                  IF DoWrite (rF) # Done THEN
                     WinPostMsg (ChildFrameWindow, WM_PAD,
                        MPFROM2SHORT (PAD_ErrWrtFile, 0),
                        ADR (rFname));
                     ErrorQuit;
                  END;
                  trys := 1;
                  SendAck (rSeq);
               ELSE
                  INC (trys);
                  IF trys = MAXtrys THEN
                     WinPostMsg (ChildFrameWindow, WM_PAD,
                        MPFROM2SHORT (PAD_ExcessiveErrors, 0),
                        MPFROM2SHORT (0, 0));
                     ErrorQuit;
                  ELSE
                     SendNak;
                  END;
               END;
            ELSE
               INC (trys);
               IF trys = MAXtrys THEN
                  WinPostMsg (ChildFrameWindow, WM_PAD,
                     MPFROM2SHORT (PAD_ExcessiveErrors, 0),
                     MPFROM2SHORT (0, 0));
                  ErrorQuit;
               ELSE
                  SendNak;
               END;
            END;
         END;
      END;

      NormalQuit;
   END Receive;


BEGIN   (* initialisation du module *)
   yourEOL := ASCII.cr;
   yourNPAD := 0;
   yourPADC := 0C;
END PAD.
```

DEBUG Magazine

MARCHE PC

LA PUBLICITE A DES PRIX ABORDABLES POUR TOUS !!!

Vous avez des produits ou services qui méritent d'être connus.
Vous voulez les promouvoir par des campagnes de publicité (achat ou vente).
Vous cherchez des rubriques spécifiques, claires, pratiques et faciles à trouver par vos prospects .
Vous désirez optimiser au maximum votre investissement.
Vous manifestez l'enthousiasme de réaliser des opérations suivies et répétées dans l'année.
Vous voulez un marché unique en France avec une transparence des prix incontestable.
Mais, vous ne disposez pas de budgets faramineux.

VOTRE SOLUTION : LES RUBRIQUES « MARCHE PC » !!!

1/2 PAGE : 3.950 F. HT

Marché PC de DEBUG Magazine, l'emplacement de choix pour trouver les produits référencés.

Renseignez-vous au 42.02.29.16

Demandez le Service Commercial

1/16 PAGE : 490 F. HT

TARIFS1991

1 PAGE : 7.900 F. HT
1/2 PAGE : 3.950 F. HT
1/4 PAGE : 1.980 F. HT
1/8 PAGE : 990 F. HT
1/16 PAGE : 490 F. HT

CONDITIONS DE VENTE

1. Couleurs du support + 20 %
2. Frais techniques en sus : forfait 500 F. HT
3. Paiement à la commande.
4. Insertion possible jusqu'à 10 jours avant parution.

1/8 PAGE : 990 F. HT

DEGRESSIFS

5 Parutions par an : 5 %
10 Parutions par an : 10 %
15 Parutions par an : 15 %
20 Parutions par an : 20 %

1/4PAGE : 1.980 F. HT

ADA POUR LES PROGRAMMEURS EN PASCAL

par ***K. N. King,***
traduit par ***Philippe Dalléas***

Le langage Pascal est à la base de la définition d'Ada et, partant, les programmes en Ada ont un aspect semblable à ceux en Pascal. Ada, toutefois, n'est pas un sur-ensemble de Pascal et, comme nous le verrons plus loin, il lui manque au moins deux propriétés du Pascal (incidemment, quand je fais allusion au Pascal, il s'agit du Pascal Standard ISO. Certains compilateurs Pascal contiennent quelques extensions à la manière d'Ada).

Bien qu'Ada soit fondé sur le Pascal, il s'agit d'un langage bien plus étendu. Non seulement Ada a-t-il de nombreuses propriétés, mais encore peut-on les combiner entre elles de façon virtuellement illimitée. Se mettre à Ada est assez facile à cause de ses ressemblances formelles avec le Pascal. En acquérir la maîtrise est bien plus ardu de par sa taille et sa complexité.

Comme Pascal, Ada offre un large éventail de types de données, dont les tableaux, enregistrements et pointeurs (les types « pointeur » s'appellent « accès » en Ada), à la différence que les types « tableau » et « enregistrement » d'Ada connaissent une option complémentaire inexistante en Pascal. Les structures conditionnelles d'Ada (les commandes if, case, while et for) sont comme en Pascal. A la place de la commande repeat de Pascal, Ada dispose de la commande loop, d'un emploi plus général. Comme Pascal, Ada inclut les notions de procédure et de fonction, collectivement connues en tant que sous-programmes.

De même que Pascal, Ada est un langage fortement caractérisé. Chaque objet doit avoir une nature fixe déclarée, et, en général, lorsqu'elles sont combinées dans une expression, les natures doivent être homogènes. Pascal a de nombreuses exceptions au principe de caractérisation rigoureuse. En particulier, Pascal permet de mêler des valeurs de nature réelle et entière dans les expressions ; une valeur de nature entière est automatiquement convertie en type real avant d'être combinée à une valeur de nature réelle. Ada est plus rigoureux que Pascal ; les valeurs de types integer et float (qui correspond, en Pascal, au type real) ne peuvent être combinées dans une expression en l'absence d'une explicite conversion de nature.

Se mettre à ADA est assez facile, à cause de ses ressemblances formelles au Pascal. Par contre, dominer ADA est bien plus ardu, de par sa taille et sa complexité.

sécurité du Pascal, Ada associe la puissance du langage C. Les compilateurs Ada sont indispensables, lorsqu'on gère une grande variété de types, afin de détecter les erreurs potentielles dès la compilation. En outre, les programmeurs peuvent outrepasser le contrôle des natures s'il empêche d'accéder à des mécanismes sous-jacents.

Dans cet article, nous analyserons quelques unes des plus significatives propriétés d'Ada, y compris les notions de polysémie et de paquet, et les types privé, exception et générique. Le manque de place empêchant l'examen détaillé de quelques propriétés importantes, dont la définition de tâches et de représentations, nous allons les résumer à l'instant.

A la différence de Pascal et de C (et de beaucoup d'autres langages courants, en la matière), Ada supporte en soi la notion de tâche. Un programme Ada peut consister en plusieurs tâches, chacune s'effectuant - du moins sous l'angle de vue du programmeur - de manière asynchrone et en parallèle. Ada dispose d'un mécanisme inhabituel (le rendez-vous) afin que les tâches se synchronisent entre elles et se communiquent des informations ; les tâches peuvent aussi communiquer entre elles via des variables partagées. Les propriétés des tâches Ada comprennent le support d'applications « temps-réel ».

Comme Pascal, Ada libère le programmeur des détails du niveau machine. Mais au contraire de Pascal, Ada permet d'accéder à ces détails quand c'est indispensable. Par exemple, les programmes Ada peuvent situer une variable à une adresse particulière ou préciser sa taille, exécuter des conversions de types non vérifiés, ou faire une foule d'autres choses dépendant des particularités de la machine cible (la machine où le programme devra tourner) ou de celles du système Ada qu'on utilise.

Etonnamment, pour un langage d'une telle richesse, Ada est dépourvu de deux propriétés du Pascal : les ensembles et l'aptitude à passer un sous-programme, pris en tant que paramètre, à un autre sous-programme. Le support des ensembles n'est pas aussi précieux en Ada qu'en Pascal parce que les programmeurs Ada peuvent toujours créer un paquet donnant un type d'ensemble et une gamme d'opérations d'ensemble.

L'avantage de l'approche « faites-le vous-même » d'Ada réside en ce qu'on peut adapter le type d'ensemble à ses besoins - en l'espèce, on peut décider de la taille maximale de l'ensemble, comme de la méthode selon laquelle il sera intégré.

Pascal vs. Ada : un exemple

Quand on s'en sert pour écrire des programmes simples, Ada ne diffère guère de Pascal. Toutefois, il faut garder présentes à l'esprit quelques caractéristiques d'Ada, spécialement lorsque l'on est un programmeur rompu au langage Pascal. D'une part, les procédures d'entrée/sortie ne sont pas construites dans Ada comme elles le sont en Pascal. Pour effectuer des E/S, un programme en Ada peut se servir de l'un des paquets standards d'E/S fournis avec chaque système Ada.

De ces paquets E/S d'Ada, le plus souvent utilisé est Text_IO, lequel procure les sous-programmes de lecture et d'écriture d'informations textuelles. Text_IO comporte les sous-programmes Get, Put, Skip_Line et New_Line. Get lit un élément de données, tandis que Put l'écrit (comme nous le verrons plus loin, Get et Put sont « surchargés » ; Texte_IO est actuellement une collection de sous-programmes ayant ces noms). Skip_Line saute les caractères résiduels de la ligne en cours de lecture. New_Line termine la ligne en cours d'écriture et va à la suivante.

Pour qu'un programme puisse accéder aux sous-programmes composant Text_IO, il lui faut désigner la bibliothèque par une proposition clause. D'elle-même, une telle proposition donne accès aux sous- programmes composant Text_IO, mais chaque appel d'un sous-programme doit désigner le paquet. Par exemple, un appel de Put peut être de la forme :

```
Text_IO.Put(Ch);
```

Faire suivre une proposition clause d'une proposition with donne accès direct aux noms dans Text_IO. Si une proposition use existe, les appels de Put n'ont pas à désigner Text_IO :

```
Put(Ch);
```

En Ada, le programme principal peut aussi bien être une procédure qu'une fonction (c'est, habituellement, le premier programme). Les noms des procédures, comme tout autre nom, en Ada, peuvent contenir le trait de soulignement. Comme Pascal, Ada ne distingue pas les majuscules des minuscules dans les noms.

Le signe point-virgule clôt une commande Ada ; il doit figurer en fin de chaque commande. En Pascal, ce signe est plus un séparateur de commandes qu'une terminaison. Les chaînes de caractères sont délimitées par des guillemets en Ada, tandis qu'en Pascal elles le sont par des apostrophes. Les commentaires Ada débutent par deux tirets et s'achèvent en fin de ligne.

L'exemple 1, page (51), présente un programme en Pascal dont le but est de dénombrer les occurrences de chaque caractère d'une suite lue. L'exemple 2, page (51), est un programme équivalent, rédigé en Ada.

L'exemple 2 met en évidence l'une des « ficelles »d'Ada. Les sous- programmes de lecture et d'écriture d'entiers ne se trouvent pas dans Text_IO lui-même, mais dans le paquet générique Integer_IO, nichée elle-même au sein de Text_IO. Les lignes 4 et 5 activent ce paquet et rendent accessible ses sous-programmes. Nous analyserons plus loin et de manière détaillée les paquets génériques et leur activation. Les tableaux d'Ada sont comparables à ceux de Pascal. Voyez en ligne 6, cependant, qu'Ada requiert des parenthèses, au lieu de crochets, pour les délimiter. De même, Ada se sert d'unions pour initialiser les tableaux. L'union (others => 0) stipule que tous les éléments du tableau Counts sont ramenés à la valeur 0. A la ligne 7, notez que le type « caractère » d'Ada se désigne par Character et non char, et, à la ligne 9, que l'instruction while d'Ada est de la forme while ... loop ... end loop. Le nombre d'instructions figurant entre les mots loop et end loop n'est pas limité. La fonction End_Of_File provient du paquet Text_IO.

A la ligne 11, on peut omettre les parenthèses enfermant 'a' <= Ch et Ch <= 'z', parce que l'opération and a moindre préséance que l'opération <=. Les parenthèses sont obligatoires en Pascal, car il ne connaît que 4 niveaux de préséance pour les opérations. Ada a 6 niveaux de préséance, et les opérations logiques and, or et xor ont moindre préséance que les opérations comparatives <, <=, >, >=, = et/= (non

```
program CountLetters(input, output);
   {compte les occurrences des lettres dans le flux d'entrée}
   var Counts: array ('a'..'z') of integer;
       Ch: char;
begin
   for Ch:='a' to 'z' do
      Counts [Ch] :=0;
   while not eof do
      begin
       read (Ch);
       if ('a'<= Ch) and (Ch <= 'z') then
         Counts [Ch] := Counts [Ch] + 1
       else if ('A'<= Ch) and (Ch <= 'z') then
         begin
              Ch:= chr (ord (Ch) - ord('A')+ ord ('a'));
              Counts [Ch]:= Counts [Ch] + 1
         end
       end;
    for Ch:= 'a' to 'z' do
       writeln (Ch, Counts [Ch]:6)
end.
```

Exe.1 : Programme en Pascal qui comporte les occurrences des lettres

```
1.with Text_10; use Text_10;
 2.procedure Count_Letters is
 3.  --compte les occurrences des lettres dans le flux d'entrée
 4.  package Int_10 is new Integer_10 (Integer);
 5.  use Int_10;
 6.  Counts: array ('a'..'z') of Integer := (others => 0);
 7.  Ch: Character;
 8.begin
 9.  while not End_Of_File loop;
10.    Get (Ch);
11.    if 'a'<= Ch and Ch <= 'z'then
12.       Counts (Ch) := Counts (Ch) +1;
13.    elsif 'A'<= Ch and Ch <= 'Z' then
14.       Ch := Character'Val  (Character'Pos (Ch) -
15.                             Character'Pos ('A') +
16.                             Character'Pos ('a'));
17.       Counts (Ch) := Counts (Ch) + 1;
18.    end if;
19.  end loop;
20.  for Ch in 'a'..'z' loop
21.    Put (Ch);
22.    Put (Counts (Ch), 6);
23.    New_Line;
24.  end loop;
25.end Count_Letters;
```

Exemple 2: Même exemple en Ada

égal). Dans les lignes 11-18, remarquez que l'instruction Ada if est de la forme if ... then ... {elsif ... then ...} [else ...] end if, où les accolades signifient qu'il peut y avoir tout nombre de propositions elsif (voire aucune) et, les crochets, que la proposition else est facultative.

En Ada, on peut associer des attributs aux entités (y compris les objets et les natures). L'un des attributs de la nature Character est la fonctionCharacter' Pos, utilisée aux lignes 14-16, similaire à la fonction ord en Pascal. Un autre attribut de Character est Character'Val, équivalent à la fonction chr en Pascal. En ligne 20, l'instruction for, à l'instar de while, est un cas particulier de l'instruction loop. L'instruction for détermine un paramètre d'itération (Ch, dans notre cas) et un domaine de valeurs que ce paramètre assumera ('a' .. 'z'). Un paramètre d'itération n'est pas une variable ordinaire (bien qu'il puisse en avoir le même nom) : il n'y a pas à le déclarer et il n'est pas connu hors l'itération. Dans l'appel de Put (ligne 22), le second paramètre indique que Counts(Ch) s'écrit avec au moins 6 caractères; si Counts(Ch) est de moins de 6 caractères, il sera précédé d'espaces en suffisance. Ce paramètre est un « paramètre par défaut » : si on l'omet, il prend une valeur par défaut. Les sous-programmes Ada peuvent avoir une quantité de « paramètres par défaut ».

La polysémie

Une des plus originales propriétés d'Ada est d'accepter les polysémies (1). Le Pascal a en soi quelques polysémies (tel le symbole +, qui représente l'addition d'entiers, l'addition de réels et l'union d'ensembles), mais ne fournit pas au programmeur le moyen d'en définir. Ada permet à différentes sortes d'entités, y compris les noms de sous-programmes, d'être des polysémies. Lorsqu'il rencontre un appel à un sous-programme polysémique, un compilateur Ada doit faire le choix de la variante. Il construit sa décision en fonction du nombre de paramètres et de leurs natures.

Les procédures Get et Put de Text_IO sont fortement polysémiques. Par exemple, examinez la procédure Put montrée dans l'exemple 3. Si l'appel à Put se fait avec 2 paramètres, le premier étant de type File_Type, et, le second, de type Character, alors le compilateur conclut-il que l'appel concerne la seconde forme de Put.

L'un des avantages de la polysémie est que des sous-programmes comparables peuvent être désignés par un même nom, et que, de la sorte, les programmeurs ont à se souvenir d'un moindre nombre de mots. A l'inverse, si on en abuse, la polysémie peut rendre les programmes difficiles à comprendre.

```
procedure Put (File: File_Type; Item:
Character);
procedure Put (Item: Character);
procedure Put (File: File_Type; Item: String);
procedure Put (Item: String);
```

Exemple 3: Surchargement de la procédure PUT

Les paquets

Un programme Ada se compose d'un ou plusieurs éléments compilés. L'un d'eux doit faire fonction de programme principal ; les autres sont soit des sous-programmes, soit des paquets. Un paquet est une collection d'autres entités Ada, dont les constantes, les types, les variables et les sous-programmes. Chaque paquet se décompose en deux parties : la définition et le corps. La définition établit la liste des informations mises à la disposition du reste du programme, et le corps contient les informations qui demeurent locales.

```
with Text_10, Length_Conversions;
use Text_10, Length_Conversions;
procedure Convert_To_Meters is
  package Int_10 is new Integer_10 (Integer);
  use Int_10;
  Feet: Integer;
begin
  Put ("Entrez une longueur en pieds:");
  Get (Feet);
  Skip_Line;
  Put ("La longueur en mètres est:");
  Put (Integer (Float (Feet)*Feet_To_Meters), 1);
  New_Line;
end Convert_To_Meters;
```

Ex.emple 5 : Un programme utilisant l'ensemble Lenght _ Conversions

Un paquet peut n'être rien de plus qu'un ensemble de déclarations d'objets et de types. L'exemple 4, montre un paquet, du nom de Length_Conversions, contenant la déclaration de quatre constantes fort utiles. Ce paquet n'a pas besoin de corps ; il n'y a pas besoin d'informations locales.

```
package Length_Conversions is
  Feet_To_Meters: constant := 0.3048;
  Inches_To_Centimeters: constant := 2.54;
  Miles_To_Kilometers: constant := 1.6093;
  Yards_To_Meters: constant :=0.9144;
end Length_Conversions;
```

Exemple 4 : La spécification de l'ensemble Lenght _ Conversions

Les entités décrites dans la partie des définitions d'un paquet sont à la disposition des autres unités de compilation (elles en sont « clientes »). Toutefois, premièrement, chaque cliente doit mentionner le paquet via une proposition with. L'exemple 5, montre un programme, du nom de Convert_To_Meters, utilisant des constantes se trouvant dans Length_Conversions. Notez que Convert_To_Meters accède aux constantes se trouvant dans Length_Conversions de la même manière qu'il accède aux sous_programmes dans Text_IO : via les propositions with et use.

De même, une collection de sous-programmes référencés constitue-t-elle, par nature, un paquet. L'exemple 6, page, montre la description d'un paquet (Angle_Conversions) fournissant les fonctions directe et réciproque de conversion degrés - radians. Quant un paquet fournit des sous-programmes, seuls les entêtes des sous-programmes apparaissent dans la description du paquet. Les clients du paquet ont besoin de savoir comment appeler les sous-programmes, et non pas comment ils fonctionnent.

Le paquet Angle_Conversions, à la différence de Length_Conversions, doit avoir un corps. Le corps contient toutes les déclarations des sous-programmes du paquet. L'exemple 7, page (53), montre le corps d'Angle_Conversions. Remarquez qu'une fonction Ada s'achève par l'exécution d'une instruction return; l'expression suivant le mot return contient la valeur que la fonction renvoie.

Un paquet peut contenir des structures locales (1) de données. Pour montrer comment de telles structures peuvent être établies, nous allons rédiger un petit programme dont le but et d'inverser l'ordre des caractères d'une chaîne saisie par l'opérateur. C'est facile à réaliser en se servant d'une pile (structure LI-FO) : au fur et à mesure que le programme lit les caractères, il les empile ; quand ils ont été tous lus, il les retire un à un de la pile et les écrit dans l'ordre où ils viennent. Cela va nous permettre d'écrire un paquet offrant l'accès à des sous-programmes (les actions sur la pile) tandis que la pile restera locale au corps du paquet. Nous avons besoin de trois actions portant sur la pile : Push (empiler un caractère), Pop (prendre un caractère sur la pile) et Is_Empty (dit si la pile est vide). L'exemple 8, page (53), montre un paquet, appelé Char_Stack, fournissant ces actions. Un paquet de gestion de pile plus complet pourrait inclure d'autres actions (trouver l'élément sommet de la pile sans l'enlever, dire quand la pile est pleine, etc.); pour simplifier, nous nous contenterons de trois actions.

L'exemple 8 montre un aspect d'Ada qu'on n'a pas vu dans les exemples précédents. Au lieu des paramètres de valeur et de variable (var) du Pascal, Ada offre trois modes de paramètres : in, out et in out. Un paramètre formel est dit avoir le mode in s'il représente une valeur à passer au sous-programme ; il ne peut être altéré par le sous-programme. Un paramètre en mode out représente une variable que le sous-programme chargera d'une valeur. Un paramètre en mode in out représente une variable que le sous-programme lira et altérera. Le mode par défaut est in. Le paramètre X passé à Push a le mode in parce que Push a besoin du contenu de X, et non de le modifier. Dans Pop, le paramètre X a le mode out pour que Pop puisse y charger la valeur prélevée sur la pile.

```
package Angle_Conversions is
   function Degrees_To_Radians (Degrees: Float) return Float;
   function Radians_To_Degrees (Radians: Float) return Float;
end Angle_Conversions;
```

Exemple 6 : La spécification de l'ensemble Angle _ Conversions

```
package body Angle_Conversions is

  Two_Pi: constant:= 2.0 * 3.14159;

  function Degrees_To_Radians (Degrees: Float)return Float is
  begin
     return Two_Pi * Degrees / 360.0;
  end Degrees_To_Radians;

  function Radians_To_Degrees (Radians: Float) return Float is
  begin
      return 360.0 * Radians / Two_Pi;
  end Radians_To_Degrees;

end Angle_Conversions;
```

Exemple 7. Le corps de l'ensemble Angle _ Conversions

```
package Char_Stack is

   procedure Push(X: Character);
     --empile X dans la pile

   procedure Pop(X: out Character);
     --stocke le sommet de la pile dans X, puis dépile

   function Is_Empty return Boolean;
     --retourne True si la pile est vide, False sinon

end Char_Stack;
```

Exemple 8 : La spécification de l'ensemble Char _ Stack

```
package body Char-Stack is

 Stack_Size: constant := 100; --taille maximum de la pile
 Stack_Array: array (1..Stack_Size) of Character;
 Top_Of_Stack: Integer range 0..Stack_Size := 0;

 procedure Push(X: Charcter) is
 begin
   Top_Of_Stack := Top_Of_Stack + 1
   Stack_Array (Top_Of_Stack)' := X;
 end Push;

 procedure Pop (X: out Charcter) is
 begin
   X := Stack_Array (Top_Of_Stack);
   Top_Of_Stack := Top_Of_Stack - 1;
 end Pop;

 function Is_Empty return Boolean is
 begin
   return Top_Of_Stack = 0;
 end Is_Empty;

end Char_Stack;
```

Exemple 9 : Le corps de l'ensemble Char _ Stack

Le corps de Char_Stack (cf. exemple 9, page (53) contient la description des objets rendant la pile locale, ainsi que la déclaration des sous-programmes agissant sur la pile. Notez que la valeur de Top_Of_Stack ne peut qu'être comprise entre 0 et Stack_Size; cela équivaut au sous-domaine en Pascal. Pour l'instant, ignorons que Top_Of_Stack puisse valoir plus que Stack_Size ou qu'elle soit négative ; dans un autre numéro, nous étudierons la gestion des erreurs. L'exemple 10, page (54), montre un programme principal se servant du paquet Char_Stack pour inverser une chaîne de caractères.

La compilation isolée

Les sous-programmes et paquets variés constituant un programme peuvent être conservés dans des fichiers distincts et compilés séparément, tout comme le descriptif et le corps d'un paquet. Toutefois, l'ordre de compilation des éléments compte. La partie descriptive d'un paquet doit être compilée avant le corps du paquet et avant tous les « clients » (p.ex., on doit compiler le descriptif de Char_Stack avant d'en compiler le corps, et avant de compiler Reverse_String). Cette règle fait que le compilateur puisse accéder aux informations contenues dans le descriptif du paquet lorsqu'il en compile le corps (ou les « clients »). Etre rigoureux sur l'ordre de compilation permet au compilateur Ada d'effectuer toutes sortes de vérifications sur les liaisons entre unités de compilation.

Quand le corps d'un paquet est modifié et recompilé, le reste du programme n'a pas besoin de l'être, pourvu que la partie descriptive du paquet n'ait pas été changée. Turbo Pascal 4.0 offre une propriété analogue au paquet, l'unité, mais il y manque bien des avantages des paquets d'Ada. Une unité de Turbo Pascal a une section « interface » et une section « installation », mais ces sections doivent être contenues dans le même fichier. Au bout du compte, si la section « installation » d'une unité est modifiée, tous ses « clients » devront être recompilés, même si la section « interface » n'a pas été elle-même modifiée.

Les types « privé »

Le paquet Char_Stack est un exemple de composant logiciel réutilisable. Bien qu'écrit pour être utilisé par le programme

Reverse_String, Char_Stack est assez général pour pouvoir servir à d'autres programmes. Cependant son utilité reste limitée, parce qu'il ne maintient qu'une seule pile locale de caractères. Si on écrivait un programme ayant besoin de deux piles, ou plus, il faudrait qu'on ait un type de pile permettant de déclarer autant de variables de pile qu'il serait utile.

Nous pouvons facilement modifier le paquet pour fournir un type Char_Stack repris dans les sous-programmes Push, Pop, et Is_Empty. L'exemple 11, montre les déclarations du nouveau paquet, qu'on appellera Char_Stacks. Un « client » de Char_Stacks peut déclarer plus d'une variable de type Char_stack et se servir des sous- programmes Push, Pop et Is_Empty pour agir sus ces variables :

```
S:Char_Stack;
...
Push(S,Ch);
```

Il y a une difficulté inhérente à une telle déclaration du type Char_Stack, toutefois : une variable Char_Stack est structurée comme un enregistrement, et le « client » a plein accès aux éléments de cet enregistrement. Rien ne l'empêche d'agir sur les éléments d'une variable Char_Stack. Par exemple, il peut escamoter la procédure Push ainsi :

```
S:Char_Stack;
...
   S:Top_Of_Stack:=S.Top_Of_Stack +

   S.Stack_Array(S.Top_Of_Stack):=Ch;
```

Cette pratique est dangereuse parce que rien ne vous assure que la pile soit consistante. Par exemple, le « client » peut accroître Top_Of_Stack sans mettre à jour Stack_Array, ou modifier un élément de Stack_Array sans qu'il soit au sommet de la pile.

Une autre difficulté, liée à la manipulation directe de la pile, est que le « client » devient dépendant d'une unique représentation de la pile. On préférera avoir recours à d'autres formes de représentation de la pile. Par exemple, en cas de risque de dépassement de pile, on peut modifier le paquet Char_Stacks de manière à ce qu'une variable Char_Stack soit un pointeur vers les enregistrements corrélatifs. Si les « clients » ne font que se servir des sous-programmes Push, Pop et Is_Empty pour accéder aux variables Char_Stack, on n'a pas besoin de les modifier.

```
with Text_10, Char_Stack;
use Text_10, Char_Stack;
procedure Reverse_String is
   Ch: Character;
begin
   Put("Entrer la ligne à inverser:");
   while not End_Of_Line loop
       Get (Ch);
       Push (Ch);
   end loop;
   Skip_Line;

   Put("L'ordre inverse est: ");
   while not Is_Empty loop
       Pop (Ch);
       Put (Ch);
   end loop;
   New_Line;
end Reverse_String;
```

Exemple 10 : Programme utilisant l'ensemble Char _ Stack

```
package Char_Stacks is

 Stack_Size: constant := 100;
 type Array_Type is array (1..Stack_Size) of Character;
 type Char_Stack is
       record
         Stack_Array: Array_Type;
         Top_Of_Stack: Integer range 0..Stack_Size := 0;
 end record;

procedure Push (S: in out Char_Stack; X: Character);
  --empile X sur la pile

procedure Pop (S: in out Char_Stack; X: out Character);
  --stocke le sommet de la pile dans X, puis dépile.

function Is_Empty (S: Char_Stack) return Boolean;
  --retourne True si la pile est vide, False sinon

end Char_Stacks;
```

Ex. 11: La spécification d'un ensemble Char- _ Stacks; Chars _ Stack est de type ordinaire

Les concepteurs d'Ada se sont rendu compte des difficultés pouvant survenir quand les « clients » accèdent à la représentation d'un type tel Char_Stack. Pour empêcher de tels accès, on peut déclarer Char_Stack en tant que de type private. Dans ce cas, les « clients » ne peuvent pas profiter du fait que la variable Char_Stack soit en réalité un enregistrement ; ils peuvent fournir une variable Char_Stack à un appel de Push, Pop ou Is_Empty, mais ne peuvent directement en examiner ou en modifier les éléments.

L'exemple 12, page (55), montre l'aspect de Char_Stacks quand on rend Char_Stack de type private. Notez que la déclaration de Char_Stack apparaît encore dans la partie descriptive du paquet, mais qu'elle est rendue « locale » enfin, dans une section connue comme la partie privative. Les informations contenues dans la partie privative des descriptions d'un paquet sont visibles du compilateur, mais non des « clients » du paquet. Ils savent seulement que Char_Stack est un type, non qu'il s'agit d'un enregistrement.

L'exemple 13, page (55), montre le corps du paquet Char_Stacks. Le corps ne doit pas dépendre de ce que Char_Stack soit ou non de type private. L'exemple 14, page (56), montre le programme Reverse_String après qu'il ait été modifié pour utiliser le paquet Char_Stacks.

Les exceptions

Une exception est une erreur (division par 0, par exemple) ou tout autre situation inhabituelle survenant au cours de

```
package Char-Stacks is

   type Char-Stack is private;

   procedure Push(S: in out Char-Stack; X: Character);
     --empile X

   procedure Pop (S: in out Char-Stack; X: out Character);
     --stocke le sommet de la pile dans X, puis dépile

   function Is_Empty (S: Char_Stack) return Boolean;
     --retourne True si la pile est vide, False sinon

private
   Stack_Size: constant := 100;
   type Array_Type is array (1..Stack_Size) of Character
   type Char-Stack is
       record
          Stack_Array: Array_Type;
          Top_Of_Stack: Integer range 0..Stack_Size := 0;
       end record;
end Char-Stacks;
```

Ex.12 : La spécification de l'ensemble Chars _ Stacks ; Chars _ S tack est de type privé

l'exécution d'un programme. Ada procure le moyen de désigner les situations erronées (déclarer des exceptions), afin d'y faire face (gestion des exceptions), ou de les provoquer (forcer une exception).

Les exceptions suivantes font partie intégrante d'Ada : Constraint_Error, Numeric_Error, Program_Error, Storage_Error, et Tasking_Error. Les plus courantes en sont Constraint_Error, qui se révèle chaque fois qu'un élément restrictif est outrepassé (tel un indice hors domaine), et Numeric_Error, qui se révèle lors de situations comme le dépassement de capacité ou la division par 0. Bien qu'il n'y ait que cinq exceptions prédéfinies, les programmeurs peuvent en définir d'autres, à leur gré.

Qu'arrive-t-il quand une exception se révèle au cours de l'exécution d'un programme ? S'il n'y est pas prévu de gérer l'exception, il s'arrête (habituellement, quand une exception n'est pas gérée, un système Ada affiche le nom de l'exception et le numéro de la ligne où elle est apparue). Afin de pouvoir gérer les exceptions, il nous faut ajouter, à un endroit particulier du programme, les instructions disant quoi faire lors de la survenue de telle exception. Voici une manière de définir une gestion d'exceptions :

```
begin
    instructions au cours desquelles
    l'exception peut se révéler
    exception
        when nom_exception => code
        de gestion de l'exception
end;
```

Si l'exception désignée se révèle au cours des instructions comprises entre les mots begin et exception, le contrôle passe au gestionnaire, qui, après exécution, rend la main à l'instruction qui suit le mot end. Si aucune exception ne s'est révélée durant l'exécution des instructions comprises entre les mots begin et exception, tout ce qu'il y a entre les mots exception et end est sauté.

Considérons la procédure Push dans le paquet Char_Stacks. Quand S.Top_Of_Stack est égal à Stack_Size, augmenter S.Top_Of_Stack va lever l'exception Constraint_Error, parce que S.Top_Of_Stack est contrainte d'appartenir au domaine [0 .. Stack_Size]. Quand une exception est levée et que le sous-programme en cours n'en prévoit pas la gestion, celui-ci rend la main au sous-programme appelant qui trouve l'exception levée. Tandis que Push n'a pas besoin de gérer le cas Constraint_Error, l'appelant de Push, lui, doit se donner le moyen de gérer l'exception. Malheureusement, le « client » doit avoir quelques informations sur la manière dont la pile a été installée, afin de savoir quelle exception peut être levée. Si la pile a été installée en tant que liste relative, le dépassement de capacité de la pile peut être signalé en tant que Storage_Error au lieu de Constraint_Error.

A cause de problèmes de cette nature, Ada prévoit la déclaration de nouvelles exceptions. Si une exception est déclarée au cours de la description d'un paquet, elle devient disponible pour les « clients » du paquet. Dans le cas du paquet Char_Stacks, deux erreurs peuvent survenir au cours de son utilisation : aller au-delà (empiler alors que la pile est pleine), et aller en-deçà (prélever alors que la pile est vide). C'est pourquoi on déclare les exceptions désignées par Overflow et Underflow dans la description de Char_Stacks (cf. exemple 16, page (57).

Les « clients »de Char_Stacks peuvent dès à présent prévoir la gestion des cas Overflow et Underflow. L'exemple 17, page (57), montre Reverse_String modifié pour gérer le cas Overflow (si la pile est en dépassement, Reverse_

```
package body Char_Stacks is

  procedure Push(S: in out Char_Stack; X: Character) is
  begin
       S.Top_Of_Stack := S.Top_Of_Stack + 1;
       S.Stack_Array (S.Top_Of_Stack) := X;
  end Push;

  procedure Pop(S: in out Char_Stack; X: out Character) is
  begin
       X := S.Stack_Array (S.Top_Of_Stack);
       S.Top_Of_Stack := S.Top_Of_Stack - 1;
  end Pop;

  function Is_Empty(S: Char_Stack)return Booelan is
  begin
       return S.Top_Of_Stack = 0;
  end Is_Empty;

end Char_Stacks;
```

Exemple 13. Le corps de l'ensemble Chars _ Stacks

String ignore tout simplement les caractères supplétifs. Notez l'utilisation de l'instruction Ada null : c'est une instruction « fantoche ».

Les génériques

Le paquet Char_Stacks est parfait pour n'empiler que des éléments de type caractère. Que faire s'il s'agit d'éléments de type entier, réel ou enregistrement ? On peut faire une copie du paquet Char_Stacks et y remplacer chaque occurrence de Character par un autre type. Ce n'est pas très satisfaisant. C'est non seulement laborieux, mais encore source de difficultés si l'on est amené à modifier le paquet (savoir, ajouter des sous-programmes, réparer une erreur, ou modifier la représentation de la pile) : il faudra rechercher toutes les copies du paquet et les mettre à jour une à une.

Les concepteurs d'Ada ont prévu ce problème et dégagé le moyen de le résoudre. Quand on écrit un paquet, Ada autorise l'omission de certaines informations : c'est lorsque le paquet sera utilisé, qu'on fournira les informations manquantes. Dans le cas du paquet Char_Stacks, on peut ne pas préciser le type des éléments de la pile; cela procure ainsi un paquet de gestion de pile totalement général, utilisable pour des éléments de tout type. Un tel paquet est dit générique.

L'exemple 18, page (59), montre les déclarations d'un paquet générique Stacks. Le mot generic, au début, détermine Stacks en tant que paquet générique. La ligne suivante précise que le type Element sera décrit ultérieurement, quand le paquet sera utilisé (Element n'est pas un type local ordinaire ; c'est un paramètre de type générique). Les mots is private signifient simplement que Stacks doit traiter Element comme étant de type local ; il ne pourrait pas, par exemple, considérer Element en tant qu'enregistrement.

Le corps de Stacks est identique au corps de Char_Stacks (exemple 16), si ce n'est que Stack remplace Char_Stack, et Element, Character.

```
with Text_10, Char_Stacks;
use Text_10, Char_Stacks;
procedure Reverse-String is
       S: Char_Stack;
       CH: Character;
begin
       Put("Entrez la chaîne à inverser:");
       while not End_Of_Line loop
         Get (Ch);
         Push (S, Ch);
       end loop;
       Skip_Line;

       Put("L'ordre inverse est:");
       while not Is_Empty (S) loop
         Pop(S, Ch);
         Put(Ch);
       end loop;
       New_Line;
end Reverse_String;
```

Ex.14. Le programme qui utilise l'ensemble Chars _ Stacks pour inverser une chaîne

Un paquet générique n'est pas un paquet courant ; c'est un gabarit à partir de quoi des paquets peuvent être créés au moyen d'un procédé appelé actualisation. L'actualisation d'un paquet générique implique qu'on remplisse les espaces qu'on avait aménagés lors de sa rédaction. Pour pouvoir se servir des entités du paquet Stacks, un client doit avant tout l'actualiser en précisant ce que sera Element.

L'exemple 19, page (59), montre comment faire dans le programme Reverse_String. Reverse_String, en premier lieu, actualise le paquet générique Stacks en indiquant que Character se substituera à Element ; le paquet résultant est dénommé Char_Stacks (tout nom convient). La clause use, à la ligne suivante, permet à Reverse_String d'accéder directement à Push, Pop et Is_Empty, sans avoir à mentionner Char_Stacks chaque fois. On a déjà rencontré l'actualisation d'un paquet générique. Les exemples 2 et 5 contiennent les lignes suivantes :

```
package Int_IO is new
Integer_IO(Integer);
use Int_IO;
```

L'objet de ces lignes est d'actualiser le paquet générique Integer_IO en indiquant que ses procédures seront utilisées pour lire et écrire des valeurs Integer (et non des Short_Integer ou des Long_Integer). Le nom du paquet actualisé est donné par le programmeur ; je me sers du nom Int_IO. La clause use, qui suit, rend les noms des procédures de Int_IO directement accessibles.

```
package Char_Stacks is

  type Char_Stack is private;

  procedure Push(S: in out Char_Stack; X: Character);
    --empile X dans la pile, augmente la taille si la pile est
      pleine

  procedure Pop(S: in out Char_Stack; X: out Character);
    --Dépile le sommet de la pile dans X
    --Si la pile est vide, elle est agrandie par le bas

  function Is_Empty (S: Char_Stack) return Boolean;
    --retourne True si la pile est vide, False sinon

  Overflow, Underflow: exception;
private
  Stack_Size: constant := 100;
  type Array_Type is array (1..Stack_Size) of Character;
  type Char_Stack is
       record
         Stack_Array: Array_Type;
         Top_Of_Stack: Integer range 0..Stack_Size := 0;
       end records;
end Char_Stacks;
```

Ex.15. La spécification de l'ensemble Chars _ Stacks avec les exceptions ajoutées

Les problèmes d'Ada

Ada n'est pas la panacée des langages de programmation. Comme beaucoup d'autres, il a ses avantages et ses inconvénients. J'ai présenté certaines propriétés intéressantes d'Ada ; jetons maintenant un coup d'oeil sur ses problèmes. Certaines puissantes fonctionnalités d'Ada cachent des inconvénients. La gestion des exceptions, par exemple, masque des erreurs ; des erreurs imprévues peuvent être manipulées à l'insu du programmeur. La polysémie et l'usage de paramètres par défaut peuvent rendre les programmes difficiles à lire.

Dès le début, des problèmes de performance, à la compilation et à l'exécution, ont été les fléaux d'Ada (il y a quelques années, j'ai ouï dire qu'un fameux compilateur Ada prenait 90 minutes pour compiler un programmes de 36 lignes, et qu'il fallait ensuite 45 minutes pour en établir les liaisons. Et cela se passait sur un VAX 11/780, alors qu'aucune autre tâche n'était active !). Les compilateurs actuels affichent une vitesse de compilation convenable, quelques uns faisant même la pige à Turbo Pascal. La lenteur de compilation est largement attribuable à certaines des plus complexes propriétés d'Ada, dont la polysémie et les génériques. C'est probablement du fait de ces propriétés, que les compilateurs Ada seront toujours plus lents que ceux du Pascal, du C ou d'autres langages répandus.

```
package body Char_Stacks is
  procedure Push(S: in out Char_Stack; X: Character) is
  begin
       if S.Top_Of_Stack = Stack_Size then
          raise Overflow;
       end if;
       S.Top_Of_Stack := S.Top_Of_Stack + 1;
       S.Stack_Array (S.Top_Of_Stack) := X;
  end Push;

  procedure Pop (S: in out Char_Stack; X: out Character) is
  begin
       if S.Top_Of_Stack = 0 then
          raise Underflow;
       end if;
       X := S.Stack_Array (S.Top_Of_Stack);
       S.Top_Of_Stack := S.Top_Of_Stack - 1;
  end Pop;

  function Is_Empty(S: Char_Stack) return Boolean is
  begin
       return S.Top_Of_Stack = 0;
  end Is_Empty;

end Char_Stacks;
```

Ex.16. Le programme utilisant l'ensemble Chars - stacks pour inverser une chaîne (avec les traitements exceptionnels ajoutés)

La mauvaise performance des programmes Ada est due à deux facteurs. Le premier est qu'Ada exige des vérifications approfondies; tandis que beaucoup de ces vérifications sont faites lors de la compilation, bien d'autres ne se feront qu'en cours d'exécution. Passer outre ces vérifications accroît la vitesse d'exécution, mais cela peut masquer des erreurs et poser des problèmes à terme. Le second facteur, ce sont les strictes exigences de validation d'Ada (voir l'encadré de la page (58), lesquelles incitent les développeurs de compilateurs à vouer bien plus leurs efforts à la validation qu'à la programmation et à l'optimisation.

Ada peut être un langage trompeur - sa réelle complexité n'apparaît pas toujours aux débutants. A cause de la similitude d'aspect d'Ada et de Pascal, les programmeurs parviennent rapidement à lire les programmes Ada. Ecrire des programmes Ada de petite ou moyenne taille est plus difficile, mais ne présente pas de sérieux obstacles à la plupart des programmeurs. Arriver à ce stade peut cependant donner aux programmeurs l'illusion de la sécurité. Maîtriser vraiment Ada est extrêmement difficile à cause de la richesse du langage, du nombre de ses propriétés (certaines rarement utilisées), et de la complexité de leurs interactions. Peu de programmeurs comprendront jamais Ada comme ils comprennent des langages moins riches, tel Pascal ou C. L'un des dangers d'Ada est que des chefs de projet, leurrés par la similitude d'Ada et de Pascal, soient amenés à croire qu'Ada n'est pas plus difficile à maîtriser que Pascal. Mais dès qu'ils auront adopté Ada, ils s'apercevront qu'aucun de leurs collaborateurs ne le connaîtra suffisamment pour faire face aux questions épineuses.

La complexité d'Ada va à l'encontre des idées des créateurs de langages de programmation qui persistent à vouloir les faire courts. Niklaus Wirth,

```
with Text_10, Char_Stacks;
use Text_10, Char_Stacks;
procedure Reverse_String is
       S : Char_Stack;
       Ch: Character;
begin
       Put("Entrez la chaîne à inverser: ");
       begin
         while not End_Of_Line loop
           Get (Ch);
           Push (S, Ch);
         end loop;
       exception
         when Overflow => null; -- ignore dépassement de capacité
       end;
       Skip_Line;

       Put("l'ordre inverse est:");
       while not Is_Empty (S) loop
         Pop(S, Ch);
         Put(Ch);
       end loop;
       New_Line;
end Reverse_String;
```

Ex.17. Le programme utilisant l'ensemble Char - S tacks pour inverser une chaîne (avec les traitements exceptionnels ajoutés)

LA VALIDATION : POINT FORT D'ADA OU CHAINON FRAGILE D'ADA ?

Parmi les objectifs du projet Ada, il y avait une ambition : réduire les coûts de développement et de maintenance des logiciels, offrir la portabilité des logiciels et des programmeurs, et renforcer les principes d'une saine ingénierie logicielle. Pour atteindre ces buts, les concepteurs d'Ada ont établi un jeu de tests de validation destinés à s'assurer que les compilateurs sont conformes au standard, tout autant que pour empêcher la survenue de sous-ensembles, sur- ensembles, ou formes dialectales d'Ada.

Théoriquement, la procédure de validation est un bon moyen pour atteindre ces buts, et il n'y a aucune raison pratique de ne pas l'installer prioritairement. Mais il y a loin de la théorie à la pratique, et il s'est avéré que c'est la procédure de validation elle- même, qui a rebuté nombre de développeurs de compilateurs, restreignant ainsi l'adoption du langage. Pour illustrer ce fait, ce qui suit est la narration des aventures d'un développeur qui n'avait pas été rebuté par la validation. (NDLR DDJ)

La validation d'Ada ou les péripéties d'un éditeur

par ***James Stewart,***
traduit par ***Ph. Dalléas***

Chez R.R. Software, nous avons procédé à environ une douzaine de validations, tant pour nos propres compilateurs que pour ceux que nous avions réalisés pour d'autres sociétés. Cette narration de la procédure de validation est faite en hommage admiratif et respectueux aux participants, qui ont sûrement autant ri que peiné.

Préparation

D'abord, faire un compilateur Ada. Bien que cela paraisse fort aisé, cela prend au développeur moyen plus de 16 années hommes et 2 millions de dollars pour y parvenir. Ensuite, obtenir de l'Administration le protocole de validation ayant cours. Sans qu'on sache pourquoi, c'est au moins aussi difficile que de faire un compilateur. La case départ, c'est le « Ada JointProgram Office » (AJPO), à Washington, lequel bureau vous renvoie aux « Ada Validation Facilities » (AVF). Les AVF vous font parvenir un « Ada Compiler Validation Capability » (ACVC) si vous envoyez le support approprié à votre ordinateur et si vous indiquez que vous envisagez une validation.

Maintenant vous êtes fin prêt pour une partie de plaisir ...

Planifier une validation

Pour valider un compilateur Ada, vous devez lier contrat avec vos AVF. Comme ils travaillent pour l'Etat, sachez que vous devrez attendre, attendre et attendre... Avant de planifier votre validation, vous avez deux choses à faire : tester votre compilateur contre l'ACVC, et vous assurer que ce dernier n'est pas périmé au moment où vous voulez valider. On supposera que votre compilateur passe tous les tests applicables (c'est un examen tout-ou-rien). Comme l'ACVC est révisé tous les ans, assurez-vous que vous êtes bien testé sur le bon modèle (actuellement, ACVC 1.10).

Si vous êtes arrivé à ce stade, faites une lettre aux AVF en vue d'un contrat de validation ; indiquez-y les systèmes d'exploitation dont vous vous servez, le numéro de version du compilateur à tester, et à quel protocole de validation vous souhaitez souscrire. Dites aussi quand vous enverrez les résultats de pré-validation, et à quelle date vous souhaitez la validation sur site. Enfin, vous donnerez aux AVF une estimation de la durée que prendra cette validation (vous en aurez une idée approximative en procédant aux tests de pré- validation).

Le contrat de validation

Ayant envoyé votre lettre aux AVF, vous en recevrez une confirmant ses termes. Vous recevrez aussi un contrat administratif, relativement court et simple, détaillant les actions des deux parties, à quelle époque la pré-validation et la validation auront lieu, et ce que cela vous coûtera. Le coût de la validation dépend autant de facteurs comme la rapidité de votre système d'exploitation, le nombre de validations objets du contrat, le nombre de pré-validations que vous souhaitez (cela diminue vos riques lors de la validation, mais c'est coûteux), que pléthore de facteurs plus subtils. Il est important de se souvenir qu'on doit payer le coût indiqué avant d'entamer la procédure de validation ; faites le chèque tout de suite si vous êtes pressé !

La procédure de pré-validation

Vous aurez préalablement testé votre compilateur chez vous contre l'ACVC ; maintenant, présentez vos résultats aux AVF, sur listing et sur support magnétique. Vos résultats seront contrôlés et l'on vous tiendra informé si l'on trouve quelque problème. Si vous contestez, la question sera soumise à « Ada Validation Office » (AVO), qui tranchera. Cette procédure peut durer de 45 jours à 3 mois ; munissez-vous d'un bon bouquin à lire pendant ce temps-là. Si votre travail a été correct, ce sera une procédure relativement rapide et facile.

La validation proprement dite

Le jour convenu arrive et votre équipe de validation se présente, prête à examiner le compilateur. Les membres de l'équipe auront besoin d'une salle pour noter les tests, ceux-ci parachevés. Ils chargeront l'ACVC et le feront fonctionner ; dès cet instant, vous suivrez leurs directives et travaillerez avec eux au moins deux fois par jour. Nos propres expériences avec les AVF Wright-Patterson se sont toujours révélées agréables; les membres de l'équipe se sont toujours montrés fort dévoués et compétents, ce qui est tout à fait secourable dans une période de tension telle celle de la validation. Quand la procédure est parachevée, ils relisent avec vous le « Validation Summary Report » final, en vous expliquant que les résultats peuvent être réexaminés et rejetés par l'AVO et l'APJO. En conséquence, attendez-vous à un délai supplémentaire pour obtenir vos certificats, soit 6 à 8 mois.

La post-validation

Vous avez maintenant le VSR définitf et vos certificats, et vous croyez que vous en avez définitivement terminé, jusqu'à ce que vous vous aperceviez que les certificats sont valables pour à peine un an. Cette procédure, qui dure 3 à 6 mois, est à renouveler tous les ans avec chaque fois, un examen plus sévère.

```
generic
  type Element is private
package Stacks is

  type Stack is private;

  procedure Push(S: in out Stack; X: Element);
    --empile X, augmente la taille si la pile est pleine

  procedure Pop(S: in out Stack; X: out Element);
    --dépile le sommet de la pile dans X
    --Si la pile est vide, elle est agrandie par le bas

  function Is_Empty(S: Stack) return Boolean;
    --retourne True, si la pile est vide, False sinon

  Overflow, Underflow : exception;

private
  Stack_Size: constant := 100;
  type Array_Type is array (1..Stack_Size) of Element;
       record
        Stack_Array: Array_Type;
        Top_Of_Stack: Integer range 0..Stack_Size := 0;
       end record;
end Stacks;
```

Exemple 18: La spécification de l'ensemble générique Stack

inventeur de Pascal et de Modula-2, en fait partie. Modula-2 apparut à peu près en même temps qu'Ada. Bien qu'il soit destiné au même type d'applications qu'Ada, Modula-2 est un langage plus simple. Il a quelques propriétés comparables à celles d'Ada, dont les paquets (modules en Modula-2), les types private (opaque, en Modula-2), la notion de tâche (intégrée dans Modula-2 en tant que coroutine), et les accès primaires à la machine sous-jacente. Mais il manque à Modula-2 bien des propriétés plus raffinées d'Ada (mais lourdes à installer), comme la polysémie, les exceptions et les génériques. En bref, Modula-2 prend l'essence d'Ada, et non son riche parfum. ■

```
with Text_10, Stacks;
use Text_10;
procedure Reverse_String is

  package Char_Stacks is new Stacks (Character);
  use Char_Stacks;

  S:  Stack;
  Ch: Character;

begin
  Put("Entrez la chaîne à inverser: ");
  begin
    while not End_Of_Line loop
       Get(Ch);
       Push(S. Ch);
    end loop;
  exception
    when Overflow => null;
  end;
  Skip_Line;

  Put("l'ordre inverse est:");
  while not Is_Empty(S)loop
       Pop(S, Ch);
       Put(Ch);
  end loop;
  New_Line;
end Reverse_String;
```

Exemple 19. Un programme utilisant l'ensemble générique Stacks pour inverser une chaîne

LES CURIEUX SONT SERVIS !

DISK EXPLORER
l'éditeur de tout support magnétique

• Editez un secteur, une piste, un cluster ou tout simplement un fichier sous forme ASCII, héxadécimale ou binaire.
• Changez les attributs des fichiers pour accéder aux données "cachées", modifiez les enregistrements verrouillés en "lecture seulement", récupérez les fichiers perdus
• Modifiez le répertoire, la FAT, l'emplacement des fichiers, les zones de synchronisation, les données, les formats des secteurs et des pistes, et même les registres du contrôleur.
• Intervenez sur des disques de tous formats (5"1/4, 3"1/2, disques durs ou virtuels) et de tous systèmes d'exploitation (MS-DOS, PC-DOS, OS/2, P-System, QNX, CP/M, PICK, PROLOGUE).
• Analysez les encodages ou les "protections", créez vos propres formats.

QUAID ANALYSER
le "must" des debuggers soft pour PC

• Visualisez toutes les interruptions générées par le programme, ainsi que leurs explications en clair.
• Guettez une zone mémoire et stoppez le programme dès que son contenu change.
• Analysez les étapes de chargement et de démarrage (Boot).
• Editez le contenu de la RAM pour vous y promener comme dans un traitement de texte.
Directement logé en mémoire protégée, **ANALYSER** suit pas à pas l'exécution d'un programme et ce, même en l'absence des sources et en dépit des "protections" contre l'analyse et le suivi.

Manuels en français, logiciels non "protégés".

BON DE COMMANDE

..... ex. du DISK EXPLORER au prix de 1012 F HT l'unité (1200 F TTC + 45 F de port)
..... ex du QUAID ANALYSER au prix de 1644 F HT l'unité (1950 F TTC + 45 F de port)
..... ex du QUAID PACK (Explorer&Analyser) 1897 F HT l'unité (2250 F TTC + 35 F de port)
❑ 5"1/4 ❑ 3"1/2
❑ Chèque joint
❑ VISA ❑ EUROCARD ❑ MASTERCARD
N° Exp
❑ Contre remboursement (+ 40 F de frais)
Nom _____
Adresse _____
_____ Tél _____
Code postal _____ Ville _____
Signature Date

COMPUMART
21, rue de Fécamp
75012 Paris

KERMIT

2 ème Partie

par Brian R. Anderson
traduction E.M. Amblard

La suite de notre portage de DOS vers OS/2

Il y a un mois, j'ai commencé un voyage qui devait nous mener du monde confortable du DOS au pays magique d'OS/2. Nous étions en compagnie d'un ami familier, Kermit, qui permet la communication entre différentes machines, qu'il s'agisse de gros systèmes ou d'ordinateurs personnels. Dans la première partie, je vous ai présenté les modules PCKermit et Shell (Listings un à neuf), ainsi que les notions de base de la programmation sous Presentation Manager. Dans la Partie II, qui comporte les listings dix à dix-neuf, nous examinons les problèmes posés par le portage de PCKermit de DOS à OS/2, puis nous étudions les possibilités de communication d'OS/2 ainsi que les entrées/sorties écran et vidéo de bas niveau. Pour finir, nous discuterons de la mise en oeuvre de l'émulation du terminal TVI950. Pour des raisons de place, nous n'avons malheureusement pas pu inclure la totalité des listings de PCKermit dans cette revue. L'ensemble complet des listings se trouve sur la disquette de Debug Magazine numéro 10 (pour plus de détails, voir l'avant dernière page du journal). Je dois également signaler que, depuis que j'ai rédigé cet article, Stony Brook a publié sa Version 2.1 de Modula-2, la première à supporter « officiellement » Presentation Manager. Le programme source mentionné ci-dessus, équivalant aux listings présentés ici du point de vue de ses fonctions, reflète les changements apportés par la nouvelle version de Modula-2.

Problèmes inattendus

Il est impossible d'écrire dans l'espace de présentation AVio pendant qu'on est en train de travailler à l'intérieur d'une tâche, à moins d'avoir créé une file d'attente de messages séparée pour la tâche. Une tentative d'accéder à AVio (ou Gpi), sans avoir auparavant créé une file d'attente séparée, fait partie des rares activités susceptibles de faire plier genoux à OS/2. A part cela, je pouvais faire tout ce que je voulais à l'intérieur d'une tâche (comme ouvrir un fichier, lire dans un fichier, envoyer au port de communication, recevoir du port de communication ou écrire dans un fichier), sans le moindre problème. Mais il suffisait que j'essaie d'envoyer des données dans l'espace de présentation AVio, pour que le système se fige, à l'exception du pointeur de la souris. Quand je déplaçais le pointeur de la souris vers la zone de l'écran où l'affichage aurait dû se produire, le pointeur disparaissait, pour réapparaître dès que je déplaçais de nouveau la souris. J'ai trouvé cela particulièrement étrange, car l'écriture dans AVio consiste essentiellement à envoyer des données dans un tampon, dont le contenu n'est affiché qu'au moment où un message WM_PAINT est détecté quand la tâche principale active la procédure de gestion des fenêtres. Plutôt que d'ajouter une deuxième file d'attente de messages, j'ai opté pour une solution consistant à écrire dans AVio seulement à l'intérieur de la tâche principale. Pour afficher leurs informations, les autres tâches doivent envoyer des messages à la tâche principale, qui les affiche ensuite dans l'espace de présentation AVio. Une limitation importante d'OS/2 réside dans le fait que les tampons des ports série sont de taille fixe, autrement dit, le programmeur d'une application n'a pas la possibilité de spécifier un tampon plus important. Comme j'ai pu le constater en mode émulation terminal (sur un IBM PS/2, Modèle 70), Presentation Manager n'est pas en mesure d'assurer le rafraîchissement de l'écran lorsque des caractères arrivent du port série à une vitesse de 960 caractères par seconde (9600 bits par seconde). Je suppose que cela est dû au gestionnaire de messages de Presentation Manager, car il n'existe aucun problème de ce genre lorsqu'on exécute un programme OS/2 similaire, mais en mode caractères. A 9600 bits par seconde, il faut environ deux secondes à l'ordinateur hôte pour envoyer un écran complet (jusqu'à 2000 caractères). Cependant, comme la procédure de fenêtre est informée de chaque caractère par l'intermédiaire d'un message (semblable au message WM_CHAR), Presentation Manager a besoin d'environ six secondes pour rafraîchir l'écran (toujours pour un IBM PS/2, Modèle 70). Avec le tampon série de taille fixe (actuellement 1Ko, mais Microsoft ne s'est pas engagé à

[SERVICE LECTEURS : INTERESSANT = 19 / MOYEN = 20/ MAUVAIS = 21]

POUR OS/2

ne pas y toucher plus tard), il y a un fort risque de dépassement de la capacité du tampon, entraînant des pertes de données et un écran brouillé. Comme je l'ai mentionné au début dans la description des modules, j'ai ajouté dans une couche une mémoire tampon supplémentaire de taille variable au module de gestion du port série de Stony Brook. Lorsque j'ai spécifié un tampon de 4Ko, le problème de dépassement de capacité avait disparu. Le port série standard d'OS/2 n'est pas le seul tampon dont la capacité risque d'être insuffisante dans la situation que je viens de décrire. La file d'attente des messages est prévue par défaut pour dix messages (ce qui est amplement suffisant pour les messages générés par les interactions de l'utilisateur, mais tout à fait inapproprié quand le port série génère 960 messages par seconde). Mais il y a pire: il « semblerait » qu'en plus de la faible valeur par défaut, il existe une limite supérieure de 1024 pour la taille de la file d'attente de messages. Au delà de cette limite, une bonne partie des interfaces clavier (y compris les raccourcis clavier) refusent tout simplement de fonctionner. Néanmoins, en prévoyant un tampon de taille importante (4Ko) pour les caractères entrants, je n'ai jamais rencontré de problèmes comme ceux que je viens de décrire pour les sections de transfert de fichiers; les modules PAD et DataLink envoient un seul message pour chaque paquet reçu ou envoyé. Même à la vitesse de transmission la plus élevée, cela équivaut à une vingtaine de messages par seconde, un débit que le système est tout à fait capable de gérer. PCKermit est capable de transférer des fichiers entre mon compatible 386 AT et mon Atari ST à une vitesse pouvant aller jusqu'à 19200 bits par seconde. OS/2 supporte certes un débit de 19200 bits par seconde, mais ce n'est pas le cas du matériel IBM PS/2, et le Modèle 70 ne fonctionnera certainement pas à 19200 bits par seconde (ni avec ce programme, ni avec aucun autre programme de communication que j'ai eu l'occasion de tester). Entre l'IBM PS/2, Modèle 70 et le gros ordinateur IBM 3083, PCKermit transfère des fichiers à une vitesse maximale de 9600 bits par seconde seulement.

Emulation de terminal

Le module TERM.MOD (Listing dix), assure l'émulation terminal TVI950. TERM.DEF fait partie des listings publiés dans la première partie de cet article, comme c'est d'ailleurs le cas de tous les fichiers de définition des modules que nous examinerons plus loin. L'émulation terminal consiste à intercepter les caractères du clavier et à les adapter au format de l'hôte (par exemple, le code de la touche Insert du clavier IBM PC n'est pas le même que le code Insert du TVI950 - le logiciel doit assurer une traduction appropriée). Les codes de contrôle en provenance de l'hôte doivent être reconnus et provoquer les actions correspondantes (dans le cas de l'émulation TVI950 par exemple, la réception d'un ESC* doit produire un effacement de l'écran). Term appelle Screen, le module qui contient véritablement les fonctions pour effacer l'écran, positionner le curseur, etc. Le module Term contient également Dir, la fonction qui valide le chemin indiqué par l'utilisateur, change le répertoire courant vers l'unité et le répertoire spécifiés et affiche ensuite le répertoire demandé. La section utilisée pour l'émulation terminal (c'est-à-dire le mode Connect) s'appelle TermThrProc (ligne 1545). Aussi longtemps que la section est active, cette procédure parcours une boucle, capturant les caractères et envoyant des messages WM_TERM (ligne 1554). Si aucun caractère n'a été reçu, un appel à DosSleep restitue le restant de la tranche de temps allouée à cette tâche au système (OS/2 ne comporte pas de boucles d'attente inutiles !). Lorsque la procédure de gestion des fenêtres détecte le message WM_TERM, elle appelle PutPortChar (ligne 1566) avec le caractère obtenu à partir du mot inférieur de mp1, le premier paramètre du message. La procédure PutPortChar effectue la première moitié de l'émulation terminal TVI950. Elle reconnaît les codes TVI950 et réalise les actions correspondantes (effacement de l'écran, positionnement du curseur, ou affichage d'un caractère). Plusieurs autres procédures l'assistent dans cette tâche : Escape (ligne 1604) gère les séquences d'échappement ; Cursor (ligne 1620) positionne le curseur ; InsertMsg et MsgOn (lignes 1636 et 1654) affichent les messages d'état envoyés par l'hôte en vidéo inverse en bas de l'écran. Lorsque la procédure de fenêtre détecte un message WM_CHAR, elle appelle PutKbdChar (ligne 1457). Cette procédure, ainsi que plusieurs autres fonctions, réali-

sent le reste de l'émulation terminal TVI950. Elles convertissent la frappe de certaines touches (par exemple, les touches de fonction et les touches de déplacement de curseur) vers les codes TVI950 appropriés. Tous ces codes spéciaux sont gérés par Function (ligne 1484). La procédure Dir (ligne 1375), qui fait également partie du module Term, n'est pas liée à l'émulation terminal. En plus d'afficher un répertoire, elle permet à l'utilisateur de changer d'unité de disque ou de répertoire (on change l'unité et le répertoire en spécifiant simplement un chemin complet au moment de demander un répertoire). Cette possibilité est nécessaire dans un programme Kermit, où l'on doit seulement spécifier le nom du fichier pour envoyer celui-ci (par exemple, vous ne pouvez pas envoyer un fichier désigné par son chemin complet). La raison en est que tous les ordinateurs utilisant PCKermit ne reconnaissent pas les spécifications de répertoire du type Unix. L'essentiel du code de la procédure Dir sert à séparer, à l'intérieur du chemin, le lecteur, le sous-répertoire et le nom du fichier.

Le module Screen

SCREEN.MOD (Listing onze) utilise les fonctions Vio d'OS/2 pour contrôler l'affichage. Vio offre plusieurs attributs, en monochrome aussi bien qu'en couleurs. Les constantes définies à partir de la ligne 1694 et les variables définies à partir de la ligne 1712 contrôlent ces attributs. Les seuls attributs émulés par PCKermit sont NORMAL, HIGHLIGHT (surbrillance) et REVERSE (inversion vidéo), c'est-à-dire les attributs nécessaires pour les interactions avec le gros système qui sert d'ordinateur hôte. Il est possible affecter des couleurs différentes à ces trois attributs. PCKermit propose cinq combinaisons de couleurs. A la ligne 1837, la procédure ClrEol utilise VioScrollUp pour effacer jusqu'à la fin de la ligne, en faisant défiler une partie de l'écran vers le haut (rempli avec des cellules de caractères « espace »ayant l'attribut NORMAL). A la ligne 1961, WriteAtt utilise VioWrtNCell pour écrire un caractère avec l'attribut courant à l'emplacement du curseur. SCREEN.MOD utilise AVio pour assurer différentes fonctions vidéo, parmi lesquelles : ClrScr, ClrEol, GetXY, GotoXY et WriteAtt (qui écrit un caractère plus son attribut à l'emplacement du curseur). Charles Petzold qualifie AVio de « solution de facilité », mais dit en même temps que ces fonctions sont « trop bonnes pour être entièrement rejetées ». Je suis d'accord avec les deux affirmations : AVio est certainement plus facile d'emploi que Gpi, mais il possède tout ce qu'il faut pour assurer l'affichage à l'écran lorsqu'on émule un VDT standard. J'estime ne pas avoir à m'excuser d'avoir adopté la solution la plus facile.

Travailler avec CommPort

COMMPORT.MOD (Listing douze), est un module fourni par Stony Brook qui assure l'envoi d'un octet à la fois au port de communication COM1 ou COM2. Le module DataLink (voir Listing dix-sept) appelle ce module à plusieurs reprises. Le module original présentait de nombreux problèmes, dont le plus sérieux était qu'en réception, il attendait jusqu'à ce qu'un caractère soit lu au lieu de retourner immédiatement un état « pas de caractère disponible », grâce à quoi le système se plantait à de nombreuses occasions. J'ai beaucoup modifié ce module pour éliminer ce problème et certains autres, ainsi que pour ajouter une gestion plus souple des tampons. Pour plus de détails, voir le chapitre « Problèmes inattendus » plus haut. J'ai donc utilisé DosDevIOCtl pour tester l'état du port série et retourner des informations d'état appropriées à l'appelant. En plus, il y avait un petit problème lors de l'initialisation des bits de stop. J'ai aussi amélioré CommPort en ajoutant des tampons alloués dynamiquement. Dans la version originale, l'appel de GetChar n'allait chercher qu'un seul caractère sur le port série. Si GetChar n'est pas appelé avec une fréquence suffisante, le tampon d'OS/2 finit par déborder. Dans ma version améliorée, chaque appel de GetChar provoque le transfert de tous les caractères du tampon d'OS/2 dans un tampon local (dont la taille peut être quelconque). Même si chaque appel de GetChar ne renvoie qu'un seul caractère, le tampon d'OS/2 (trop petit) est désormais transféré dans un tampon local (dont la taille peut être spécifiée par le programmeur). Au moment de la publication de cet article, Stony Brook devrait être en train de distribuer la version 2.1 du compilateur Modula-2, qui comportera de nombreuses améliorations (y compris un module CommPort corrigé). Stony Brook a malheureusement changé l'utilisation du paramètre de message OS/2-PM, MPARAM. Selon la documentation Microsoft et la documentation de la version antérieure de Stony Brook, MPARAM est une adresse de 32 bits. La version 2.1 de Stony Brook a modifié la représentation de MPARAM pour en faire un enregistrement. La définition de cet enregistrement se trouve à la Figure 1.

```
TYPE
   MPARAM = RECORD
      CASE : INTEGER OF
            1  : B1, B2, B3, B4 : BYTE;
            2  : W1, W2 : WORD;
            3  : L : LONGINT;
      END;
END;
```

Figure 1 : *Définition du type record de MPARAM*

MPARAM (une adresse 32 bits utilisée comme paramètre d'un message) est souvent utilisée comme variable de type entier court non signé ou caractère. Microsoft fournit plusieurs macros, comme MPFROM2SHORT, pour assurer les conversions nécessaires. La version d'origine de Stony Brook comportait des macros du même genre. La nouvelle version n'utilise cependant pas de macros mais initialise un champ de l'enregistrement avant de passer le paramètre. L'exemple 1 contient un extrait de programme qui illustre cela.

Autres modules

FILES.MOD (Listing treize) gère l'accès aux fichiers disque : ouverture, lecture, écriture et fermeture. Pour l'essentiel, FILES est repris tel quel depuis la version DOS, sauf que les conflits entre noms de fichiers ont maintenant pour effet de renommer automatiquement le fichier, au lieu de demander à l'utilisateur s'il souhaite ou non écraser le fichier. Par exemple, si le nom d'un fichier NOMFICH.TYP existant est reçu, il est renommé en NOMFICH.K0. KH.DEF (Listing quatorze), KH.MOD (Listing quinze), et PCKERMIT.H (Listing seize) comportent des définitions de constantes (p.ex. IDM_KERMIT), utilisées par les ressources. Chaque article de menu, bouton de radio, icône, etc, possède un identificateur unique. Le programme Modula-2 obtient ces informations de KH.DEF (KH.MOD est vide). Les ressources elles-mêmes obtiennent leurs informations de PCKERMIT.H. Dans

```
Premières versions du Modula-2 Stony Brook (très similaire au C Microsoft)
WinPostMsg (ChildFrameWindow, WM_PAD,
        MPFROM2SHORT (PAD_ReceivePacket, 0),
        MPFROM2SHORT (PktNbr, rSeq));

Stony Brook Modula-2 v2.1

VAR
        MP1, MP2 : MPARAM;      {Nouvelles variables requises}

MP1.W1 := PAD_ReceivePacket; MP1.W2 := 0;
MP2.W1 := PktNbr; MP2.W2 := rSeq;
WinPostMsg (ChildFrameWindow, WM_PAD, MP1, MP2);
```

Exemple 1 : *Extraits de programme illustrant la manière dont la nouvelle version de Modula-2 de Stony Brook traite MPARAM.*

un programme en C, le programme et les ressources obtiendraient leurs informations du fichier en-tête (.H), mais comme le compilateur de Modula-2 n'est pas capable de lire les fichiers .H, ces informations doivent être dupliquées dans un fichier .DEF. PAD.MOD est un assembleur / désassembleur de paquets. A cause de sa taille, le listing PAD.MOD n'est pas inclus dans cet article (mais il est disponible dans la disquette Debug Magazine). Sauf en ce qui concerne l'affichage à l'écran (par passage de messages) pour assurer l'information de l'utilisateur, et l'envoi de fichiers multiples, ce module est pour l'essentiel inchangé par rapport à la version DOS d'origine (voir « Kermit rencontre Modula-2 », Debug Magazine Janvier 1991). Pendant le transfert du fichier par Kermit, PAD regroupe les données par paquets pour les passer ensuite au module DataLink. Les affichages standard sur l'écran du terminal des messages d'information sont remplacés par l'envoi de messages OS/2-PM par la fonction WinPostMsg(), et toutes les routines d'affichages sont modifiées en une fonction unique. Cette organisation étonnante est nécessaire parce que la plupart des modules PAD sont exécutés dans une tâche sans file d'attente de messages, donc incapable de réaliser les entrées - sorties. La fonction qui rassemble les routines d'affichage est appelée à partir de la tâche principale, qui, elle, possède une file d'attente de messages. DATALINK.MOD récupère les paquets que PAD lui envoie, leur ajoute SOH et CheckSum avant de les transmettre. Le travail inverse est également réalisé. Ce module est virtuellement inchangé par rapport à la version DOS. PCKERMIT.RC (Listing dix-huit) est le fichier ressources contenant l'ensemble des ressources, sauf les bitmaps. Les ressources comportent les menus, les boîtes de dialogue et les raccourcis clavier. Les bitmaps, comprenant les icônes, les polices de caractères et divers dessins, se trouvent dans des fichiers séparés. Un fichier bitmap, appelé PCKERMIT.ICO (disponible sur la disquette Debug Magazine) contient l'icône (un dessin de Kermit la grenouille) qui s'affiche quand on réduit le programme. Les icônes sont créées de manière interactive, à l'aide d'un programme nommé ICONEDIT, qui est pour l'essentiel un programme de dessin. J'ai créé l'ensemble chaque boîte de dialogue séparément à l'aide du programme DLGBOX de Microsoft, puis j'ai fusionné l'ensemble des boîtes dans un fichier PCKERMIT.RC. J'ai ensuite ajouté les menus et les tables de raccourcis manuellement. Le compilateur de ressources convertit le fichier .RC en un fichier .RES, qui sera ensuite relié au fichier .EXE. PCKERMIT.EDF (Listing dix-neuf), est le fichier de définition .EXE. Il contient des informations pour l'éditeur de liens (p.ex. la taille de la pile, les procédures de gestion des fenêtres exportées, les procédures importées par l'intermédiaire de DLL, ainsi que le mode). Dans un projet en C, ce fichier se serait appelé PCKERMIT.DEF.

Conclusion

Pour vous familiariser avec le kernel d'OS/2 et Presentation Manager OS/2, il vous sera demandé un effort d'apprentissage important, mais le jeu en vaut la chandelle. Lorsque la structure du programme est en place, l'ajout des menus, des raccourcis clavier, de boîtes de messages d'alerte et de boîtes de dialogue ne pose plus de problème. Une fois débarrassé de ces problèmes secondaires, vous pouvez vous concentrer sur l'application elle-même. Sauf en ce qui concerne l'entrée / sortie de la console, fortement influencée par Presentation Manager, la plupart des applications OS/2-PM sont les mêmes que dans un environnement plus traditionnel. Le présent article n'aborde pas les fonctions Gpi, et je pense que cela nécessitera un nouvel effort d'apprentissage, ô combien douloureux. Mais là encore, je pense que vous ne le regretterez pas. Il est certainement plus facile d'apprendre à se servir des fonctions de Presentation Manager que d'écrire des fonctions de ce genre pour votre propre programme. Pour finir, je tiens à signaler que, depuis que j'ai rédigé cet article, j'ai continué à améliorer PCKermit. La version améliorée est disponible directement à mon adresse figurant à la fin de l'article. PCKermit est capable de transférer des fichiers entre mon compatible 386 AT et mon Atari ST à une vitesse pouvant aller jusqu'à 19200 bits par seconde.

Brian est instructeur au département informatique de l'Institut de Technologie de Colombie Britannique. On peut le joindre à l'adresse suivante :
3700 Willington Ave,
Burnaby, B.C. British Columbia,
V5G 3H2.
Canada

KERMIT POUR OS/2 :
copyright **Dr. Dobb's Journal,**
M&T Publishing Inc.
Redwood City,
CA USA 1990.
All right reserved.

TurboPascal 6.0:

En décidant de publier ce banc d'essai, je cours un grand risque. Il est, en effet, certain qu'après l'avoir lu, mes fidèles lecteurs viendront m'étriper en m'accusant de toucher des pots-de-vin chez Borland. Mais, la liberté de la presse ne souffrant aucune concession, je maintiens que TurboPascal 6.0 (TP6) est le meilleur logiciel de l'année.

Philippe KAHN, Président de Borland International

[SERVICE LECTEURS : INTERESSANT = 22 / MOYEN = 23 / MAUVAIS = 24]

Le compilateur est livré, comme c'est la coutume chez Borland, en deux configurations,dénommées TurboPascal 6.0 et Turbo Pascal 6.0 Professionnal, cette dernière contenant en plus le débogueur, l'assembleur et le profileur. Dans les deux cas, le compilateur est livré en deux versions (environnement de développement et version non interactive). Une troisième version est livrée dans la configuration Professionnal : il s'agit du compilateur non-interactif modifié pour fonctionner en mode protégé. Elle permet de compiler sous DOS des applications de très grande taille. L'environnement de développement a subi un remaniement complet, pour adopter la même présentation que celle de TurboC++.

L'éditeur, pièce centrale de cet environnement, est désormais capable de travailler sur plusieurs fichiers dans des fenêtres multiples, sait utiliser la souris, et dispose d'un langage macro permettant de le configurer à volonté. La limitation à 64K de la taille des fichiers, qui handicapait lourdement les éditeurs précédents, a été levée : l'environnement intégré accepte désormais des fichiers jusqu'à 1Mo. La configuration de l'environnement (avec les fenêtres et les fichiers ouverts) peut être automatiquement sauvegardée entre deux sessions de travail - il suffit de relancer l'environnement pour le retrouver dans l'état dans lequel vous l'avez quitté la veille au soir. Le langage lui-même bénéficie de plusieurs extensions. La plus intéressante est sans soute l'incorporation d'un assembleur au sein du compilateur : l'infâme inline() avec son chapelet de codes hexadécimaux peut être remplacé par du code source assembleur :

```
PROCEDURE Cap(Var Str:String); ASSEMBLER;
        ASM
        les di, Str
        mov cl,es : [di]
        inc cl
        ...
```

Cet assembleur intégré est même en mesure d'assembler les instructions spécifiques du 80286. Et si, malgré tout, vous restez un inconditionnel du TurboAssembler, sachez que vos routines externes peuvent désormais déclarer des segments de données initialisées (ou, en d'autres termes, importer des constantes typées dans les routines en assembleur). La syntaxe du langage a été étendue avec la directive {$X+}, qui autorise l'appel de fonctions d'une façon semblable au C, à savoir que le résultat de la fonction peut être ignoré. Le bout de code qui suit devient donc parfaitement légal :

```
{$X+}
Function Foo(In:Byte):Word;
  Begin
  ...
  End;

Begin
  ...
  Foo($FF);
  ...
End.
```

Une autre directive a fait son apparition. Il s'agit de {$G+}. Elle permet de générer les instructions spécifiques du 80286: ENTER, LEAVE, PUSH immédiat et instructions IMUL, SHL et SHR étendues. La déclaration des objets s'enrichit d'un nouveau mot-clé, PRIVATE, qui peut s'appliquer aux variables comme aux méthodes. Il permet de renforcer l'encapsulation des données et du code en limitant l'accès aux données et méthodes privées (autrement dit, la portée de leur définition) seulement à l'unité dans laquelle elles sont déclarées.

Enfin, les possibilités du débogueur intégré ont été étendues par l'adjonction de points d'arrêt conditionnels ainsi que de la possibilité d'examiner les registres du processeur.

Tout cela est quand même trop beau pour être vrai et effectivement, il y a quelques (petits) points noirs. Un détail mineur est la suppression de FreeMin et FreePtr (due à la modification de la gestion du tas). Les programmes qui dépendaient de ces deux variables devront être partiellement modifiés. Il s'agit principalement des programmes qui, pour une raison ou une autre compactaient le tas (avant un shell vers le DOS, par exemple). Heureusement, le nouveau gestionnaire du tas facilite beaucoup la tâche.

Mais le plus grave est sans doute le changement, une fois de plus, du format des unités compilées. Là, je trouve que Borland dépasse un peu les bornes. Depuis la version 4.0, j'ai du recompiler trois fois toutes mes unités. Cela représente plus de 10 MB de code source à compiler (et à vérifier) avec, à chaque fois, 2 à 3 jours de travail. Et encore, je dispose du source de toutes les unités. Mais que doit faire celui qui ne dispose que des fichiers compilés ? Racheter une nouvelle version de la librairie, prier pour que le concepteur fasse une mise à jour pas trop onéreuse ou aller se faire voir ailleurs ? Est-il réellement si difficile de créer un utilitaire qui permettrait de convertir les fichiers TPU sans avoir à les recompiler ?

En ce qui concerne les performances, il était difficile de faire mieux que le TurboPascal5.5 et les tests ne laissent pas apparaître de différences significatives, que ce soit lors de la compilation (tous les temps sont exprimés en secondes):

	TP5.5	TP6.0	
Savage87		1.12s	1.17s
Savage	1.06s	1.10s	
Sieve	1.05s	1.05s	

ou de l'exécution:

	TP5.5	TP6.0	
Savage87		6.23s	6.23s
Savage	31.28s	32.17s	
Sieve	37.88s	37.37s	

Peut-on faire mieux ?

Sur une application réelle (gestion de fichiers interactive; env. 5000 lignes de code source) la différence n'est guère plus importante. Seule, la directive $G+ permet de réduire (et encore...) la taille de l'exécutable:

	TP5.5	TP6.0G-	TP6.0G+
pil.	27.28s	27.33s	26.68s
e	116103	118943	114673

La grande innovation est, en fait, moins le compilateur lui-même, que la librairie d'objets qui l'accompagne. Vous m'avez souvent entendu ricaner en mon for intérieur (et même rigoler aux éclats...) lorsque d'aucuns (principalement les concepteurs et vendeurs de langages de programmation) s'extasiaient sur les avantages de la programmation Objet et notamment sur la réutilisation du code existant. Mais quel code? Jusqu'à aujourd'hui, le programmeur moyen pouvait toujours se brosser pour réutiliser des Objets, pour la bonne raison qu'il n'en existait pas. On en était toujours réduit à réinventer la roue et, ce qui plus est, à la réinventer en utilisant une technologie nouvelle (avec toutes les inconnues que cela comporte). Si le progrès, c'est ça, alors je retourne au Moyen-Age !

Mais voici que la situation a changé. Dans la boîte du TurboPascal6.0, il y a tout un troupeau d'objets. Une hiérarchie, pour employer le jargon officiel. Et je peux vous garantir, que cette hiérarchie-là, vous allez en user ! Vous avez à votre disposition, là, au bout de votre clavier, tout ce qu'il faut pour construire une interface utilisateur cohérente, avec menus déroulants, fenêtres et boîtes de dialogue, en moins de temps qu'il n'en faut pour le dire. Cette interface, vous la connaissez bien, c'est celle du compilateur lui-même. De plus, elle est entièrement testée, puisque l'environnement intégré de développement a été écrit en utilisant ces objets.

Avec Turbo Vision, votre application devient un objet. Cet objet va, en quelque sorte, être animé d'une vie propre et agir pour son propre compte. Il va s'enquérir des événements extérieurs, rafraîchir son image à l'écran au fur et à mesure de son déroulement et, de temps en temps, adresser des messages à votre code. Ces messages décrivent divers événements, tels que la pression d'une touche sur le clavier, un mouvement de la souris, le choix d'un élément du menu, l'appel d'une boîte de dialogue etc. Tout ceci se passe quasiment sans l'intervention du programmeur. Le rôle de celui-ci est de décrire les réactions de l'application à ces événements. Par contre, toute la magie noire qui permet de recueillir, de mettre en forme et d'adresser au bon endroit les messages est du ressort de TurboVision et se déroule de façon totalement transparente.

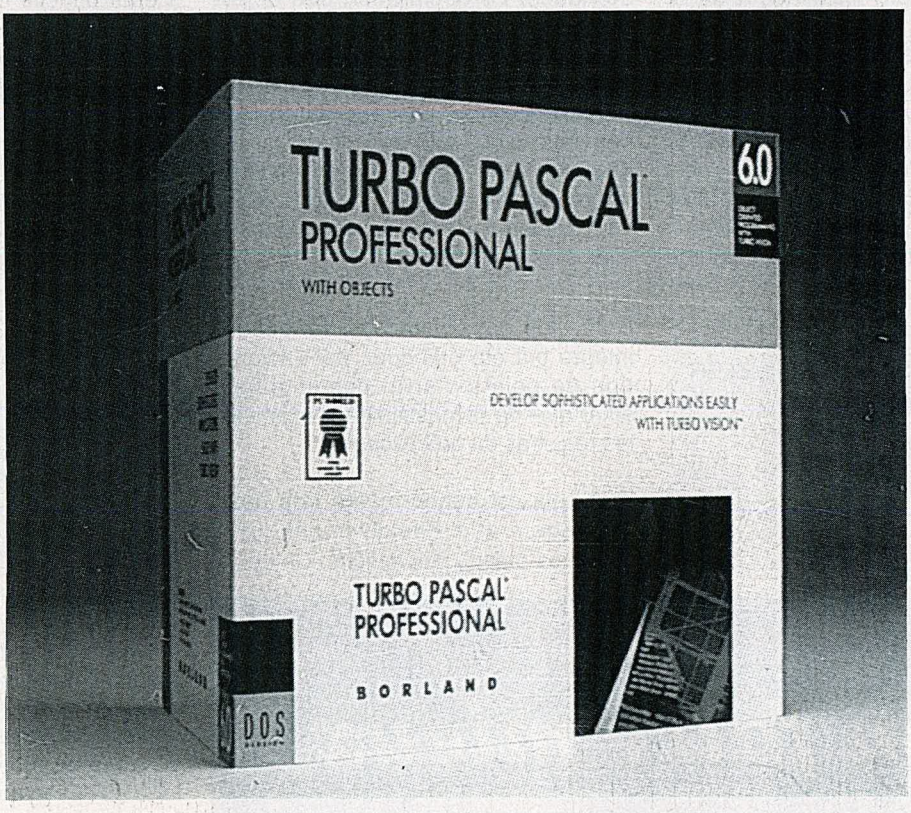
Tous les objets de la hiérarchie de TurboVision (plus de 40) ont un comportement par défaut et sont utilisables, pour la plupart, tels quels. Leurs possibilités sont particulièrement étendues. A titre indicatif, une application permettant de visualiser des fichiers texte peut être écrite en 1 heure et en moins de 200 lignes. Elle permet d'afficher le contenu de plusieurs fichiers simultanément, dans des fenêtres qui peuvent être ouvertes, déplacées, agrandies, réduites ou fermées, qui ont des barres de défilement, des menus déroulants et des boîtes de dialogues partout. Vous en connaissez beaucoup, des librairies qui permettent d'en faire autant ?

Je n'ai qu'un seul regret, à savoir que le source de TurboVision ne soit pas livré avec le compilateur. Bien que je ne sois pas, jusqu'à présent, parvenu à prendre TurboVision en défaut, ça m'interpelle quelque part au niveau du vécu de devoir construire une application sans savoir exactement ce qui se passe dans ses entrailles. C'est une situation tout à fait dans l'optique de la POO et c'est la rançon du progrès, mais j'ai quand même du mal à m'y faire. Que cependant les esprits chagrins se rassurent; tout le code de TurboVision est disponible en option avec les sources de la librairie Turbo Pascal.

En somme, alors que l'on pouvait imaginer le produit arrivé au bout de ses possibilités avec la version 5.5, TP6 repousse encore les limites de la créativité en proposant non seulement une plate-forme de développement qui, malgré quelques défauts, représente le nec plus ultra de la programmation en Pascal dans le monde DOS, mais aussi en apportant aux programmeurs, pour la première fois, des Objets réellement réutilisables.

J'ai désespérément essayé d'imaginer ce que je pourrais demander de plus et je dois avouer que les idées n'ont pas été nombreuses. TP6 aurait pu être parfait, s'il était en mesure de compiler pour Windows et si la structure des unités n'avait pas été changée. Mais je parie que les frustrés ne seront pas nombreux.

A l'heure où vous lirez ces lignes, la version française sera disponible. Ce sera un peu tard pour un cadeau de Noël, mais rien ne vous empêche de commencer 1991 avec de bonnes habitudes.



Turbo Pascal

OUF!

Suite de l'article paru dans le numéro 10 du mois de Février

COMPACTAGE ET DECOMPACTAGE DE FICHIERS : PKZIP, PKUNZIP

ORIGINE : PKWARE, Glendal, WI 53217 USA, version d'essai disponible sur le serveur RCI Calvacom BIB de la cité OUF, sur disquette OUF, et de façon générale sur tous les serveurs dont SUPTEL, coût de l'enregistrement $52.5, paiement le plus pratique avec carte VISA internationale. On reçoit alors la disquette qui contient :

* PKZIP.EXE pour compacter ;
* PKUNZIP.EXE pour décompacter ;
* MAKESFX.COM et ZIP2EXE.EXE pour créer des fichiers compactés et auto-décompactables lorsqu'ils sont lancés ;
* la documentation et un utilitaire de restauration de fichiers ZIP corrompus. Pas de procédure d'installation, il suffit de mettre les fichiers avec les autres utilitaires. C'est un logiciel indispensable, dès lors que l'on fait de la télématique, le taux de compression moyen varie selon le type de fichier texte ou binaire, de 40 à 50%, et c'est autant de temps gagné en transmission. De plus, on peut compacter en un seul fichier .ZIP plusieurs fichiers en vue d'une transmission, d'où encore gain de temps. Ce logiciel est également l'outil indispensable lorsque l'on utilise des partagiciels, le plus souvent compactés sous forme ZIP. PKZIP peut aussi rendre de précieux services lorsque l'on commence à manquer de place sur son disque dur. Pour décompacter un fichier : « PKUNZIP <filename> » puis ENTREE. Il y a toute une syntaxe expliquée dans la documentation pour traiter des situations très diverses, et en particulier pour faire le « DIR » du contenu d'un fichier zippé sans le décompacter.

INCOMPATIBILITE : aucune.

MENTION : TRES, TRES BIEN et outil indispensable de l'amateur ou du professionnel communiquant.

RENDRE MOINS DANGEREUSE LA COMMANDE DEL DU DOS : QDEL et TRASH.

ORIGINE : PKWARE (voir ci-dessus comme PKZIP), version disponible sur disquette OUF! OUFPC223 avec la lettre Hypertexte de septembre 1990, version d'essai gratuite. Au lieu de lancer « DEL <filename> » ou beaucoup plus dangereux « DEL *.* » , faire « QDEL <filename> » ou « QDEL *.* » puis ENTER, alors s'affichent les renseignements complets relatifs aux fichiers susceptibles d'être effacés ; puis se décider en répondant Y ou N à la confirmation d'effacement. Pour l'installation, il suffit de mettre ce petit fichier : QDEL.COM de 5814 octets avec les autres utilitaires.

INCOMPATIBILITE : aucune, au contraire, dans le cas des fichiers « en lecture seulement », une autre question apparaît pour le signaler et pour savoir si la demande d'effacement est maintenue.

MENTION : TRES BIEN, petit utilitaire qui amène beaucoup de sécurité à ces effacements. Il n'est plus possible de s'en passer, tant il est facile avec le « DEL » du DOS d'effacer ce que l'on ne voudrait pas. Une alternative, encore plus performante pour éviter la perte irréparable de fichiers par un « DEL » intempestif, ou un accès direct DOS ou BIOS d'un programme, consiste à utiliser la « poubelle » pour MSDOS, TRASH de Philippe Paternote (FR. 0F.), qui crée un répertoire spécial où sont sauvegardés tous les fichiers détruits. On peut à posteriori en effectuer la gestion : les détruire définitivement, les récupérer, ou les laisser en attente ...

SE DEPLACER DANS LES REPERTOIRES et VISUALISER LES EXECUTABLES : AMENU

ORIGINE : Joel Gott and The Computech Group, Providence, RI 02909 USA, version disponible avec la lettre HYPERTEXTE de juillet 1990 : OUF0790, disponible sur SUPTEL et RCI Calvacom BIB de la CITE OUF ; coût de l'enregistrement $20, versement effectué par l'intermédiaire de SOLIDARSOFT. On reçoit alors une disquette avec le logiciel disposant de toutes ses fonctionnalités.Pour l'installation il suffit de mettre le fichier AMENU.COM 6820 octets avec les utilitaires. Une fois lancé, on se promène dans les répertoires avec menu déroulant et pavé déplaçable avec les flèches. Les fichiers susceptibles d'être exécutés, selon leur appartenance, apparaissent avec des couleurs différentes : BAT, COM, EXE, et encore une autre couleur pour les sous-répertoires.La version enregistrée détecte la souris et l'utilise, et aussi, permet de lancer un fichier exécutable avec ajout d'une commande au nom du fichier.

INCOMPATIBILITE : éviter de lancer des applications sans sortir de AMENU, cela est possible, mais AMENU reste alors résident, des conflits en mémoire peuvent quelques fois apparaître. Pour sortir de AMENU appuyer sur « ESC ».

MENTION: TRES BIEN, ce petit utilitaire à l'usage se révèle très pratique.

TELEMATIQUE : PROTOCOLE DE TRANSFERT DE FICHIER : ZMODEM

ORIGINE : Omen Technologie Incorporated, Portland, Oregon 97231, version d'essai sur le serveur RCI Calvacom BIB de la CITE OUF. Seule la version enregistrée possède toutes les fonctionnalités, coût $20 payé par l'intermédiaire de SOLIDARSOFT (pas de paiement par carte VISA). On reçoit alors dans les 10 à 20 jours la version enregistrée, avec son code secret personnel. Le logiciel est largement documenté, environ 100 pages sous

OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF

forme d'un fichier texte.Il est à noter que l'on ne paie qu'une seule fois l'enregistrement. Avec le code secret, il suffit de contacter la serveur d'OMEN, pour descendre les nouvelles versions. Pour l'installation, la version enregistrée a, pour cela, un petit utilitaire qui fait le travail. On met la disquette dans le lecteur A: et l'on tape « A:INSTALL » puis ENTREE, le code secret est alors demandé. Pour utiliser ZMODEM, il suffit d'un seul fichier : DSZ.COM de 51628 octets, et aussi d'avoir défini les deux variables d'environnement : DSZLOG et DIRRX. Une syntaxe, pas très facile à comprendre, permet d'obtenir différents protocoles de transfert dont ZMODEM, le vrai YMODEM, XMODEM, KERMIT et je dois en oublier. Pratiquement je l'utilise en protocole externe de mon logiciel de communication, de la façon suivante :
* pour recevoir un fichier avec ZMODEM Moby Turbo, créer un fichier .BAT (ZMODEMD.BAT) qui ne contient que la ligne : « DSZ port 2 rz -mr » avec utilisation du port 2, et reprise de la transmission à l'endroit où l'on était en cas d'interruption. Le fichier BAT est appelé par le logiciel de communication.
* pour transmettre un fichier avec ZMODEM, créer un fichier BAT (ZMODEMU.BAT) qui contient la ligne : "DSZport2sz%1%2%3%4%5%6%7%8%9"

INCOMPATIBILITE : aucune, mais bien lire la documentation pour trouver la syntaxe adaptée, à ce que l'on veut faire, et à son logiciel de communication.

MENTION : TRES, TRES BIEN, ce protocole de transfert donne d'excellentes performances avec les serveurs PCboard ou FIDO.

TELEMATIQUE : UTILISER UN "READER of LINE" : EZ-RDR

UTILISATION : Lors de ses connections télématiques avec des serveurs du type PCboard, comme SUPTEL par exemple, il est possible de transférer, sous forme de fichier compacté .ZIP, les messages des diverses conférences choisies, et cela dans les deux sens.

Alors la connexion ne dure que quelques minutes. Puis, « off line », le READER EZ-RDR pilote le décompactage, permet la lecture des messages, en commençant par les messages personnels, permet la réponse avec reprise de certains éléments du message original, donne la liste des nouveaux fichiers disponibles sur le serveur, et donne aussi accès aux NEWS du serveur. Ceci fait, le paquet compacté des réponses est préparé par EZ et, lors de la connexion suivante, on remonte le paquet. Le serveur, lui, travaille avec le logiciel complémentaire, qui décompacte et range dans chacune des conférences choisies vos messages. C'est du grand art !

ORIGINE : Thumper Technologies, Tulsa, Oklahoma 74147-1346 USA, version d'essai disponible sous forme de disquette OUF (60F) ou sur le serveur SUPTEL. Le coût de l'enregistrement est $25, paiement très pratique par carte VISA internationale. On reçoit alors une disquette contenant la dernière version (actuellement EZ133) avec toutes les fonctionnalités, une documentation très complète sous forme de fichier texte, ainsi que son code secret. L'enregistrement n'est payé qu'une seule fois. Pour les mises à jour, il suffit de se connecter sur l'un des six serveurs qui diffusent les versions enregistrées, et avec son code secret il est possible de descendre la mise à jour. Une autre façon de procéder consiste à envoyer à Thumper Technologie : $5 (cadeau!), pour recevoir la disquette avec la nouvelle version. Il faut commencer par lire, de façon attentive, la documentation avant de faire l'installation. Il y a là une réelle difficulté, puis utiliser le programme d'installation INSTALL.EXE ; celui-ci réclame votre code secret, puis il faut configurer le logiciel avec CONFIG.EXE. Au début, il est difficile de tout comprendre, savoir que cela fonctionnera quand même et qu'il sera toujours temps, d'affiner ultérieurement. Ensuite, sur le serveur, il faut aussi régler sa configuration, ne pas stresser, on peut toujours la modifier ensuite ; puis descendre le paquet zippé. Alors "off line", lancer EZ et appeler ce paquet. EZ est vraiment un outil merveilleux, ne pas hésiter dans les messageries du serveur à poser des questions si quelque chose n'est pas compris ou ne marche pas, il y aura toujours quelqu'un pour répondre, et c'est ainsi que l'on progresse. Pour fonctionner, EZ lui-même n'a besoin que des trois fichiers : CONFIG.EXE, EZ-RDR.CFR et EZ-RDR l'exécutable. De plus, il est nécessaire de définir la variable d'environnement EZRDR et de pouvoir appeler (avec un PATH éventuel) : PKUNZIP, PKZIP et un éditeur de texte (QEDIT est tout à fait adapté). Il est de plus souhaitable de créer un sous-répertoire pour placer les paquets zippés reçus. Noter que lors des mises à jour, il suffit de remplacer le fichier : EZ-RDR.EXE, et c'est tout. L'ensemble est alors immédiatement fonctionnel.

INCOMPATIBILITES : aucune, mais attention à la confusion, EZ-RDR n'intervient à aucun moment lors de la connexion, c'est le rôle du logiciel de communication. EZ-RDR traite le paquet zippé, avant ou après cette connexion.

MENTION : EXCEPTIONNELLE !

DETECTER LES VIRUS : SCAN
LES DETRUIRE : CLEAN-UP

ORIGINE : McAfee Associates, Santa Clara, CA 95054, version complète disponible pour essai sur disquette OUF (60F)et sur pratiquement tous les serveurs (RCI, SUPTEL, réseau FIDO ...). La disquette contient une documentation complète ainsi que le mode opératoire, le logiciel distribué comporte toutes les fonctionnalités, ce qui est rare dans le domaine des partagiciels. Après la période d'essai, en principe de 1 mois, si vous voulez continuer à utiliser le logiciel, il faut verser sa participation : $25, le paiement par carte VISA est très commode. Cet enregistrement n'est à payer qu'une seule fois, la première, alors vous recevez la disquette contenant la dernière version. Pour obtenir les mises à jours, le plus simple est de les prendre sur un serveur français lorqu'elles arrivent ou par disquette OUF. Il y a en effet toujours des mordus, qui sauront se connecter sur le serveur de McAfee pour descendre la dernière version, et ils en font profiter les autres, ainsi, va et vit le partagiciel. La version 62 détecte 86 virus, cela permet de détecter les virus dans environ 95 % des cas. La version 67 en détecte 223 ! Il n'y a pas d'installation particulière, simplement lors de l'appel de SCAN il suffit de passer les paramètres nécessaires. Attention pour éviter toute corruption, faire une disquette verrouillée en écriture. Ainsi : « SCAN B: /A » puis « ENTER » va scanner tous les fichiers de la disquette dans le lecteur B:

Lorsqu'un virus est trouvé, le complément indispensable de SCAN est CLEAN-UP. CLEAN se procure dans les mêmes conditions que SCAN,

le prix de l'enregistrement est de $35. Il est également conseillé de conserver CLEAN-UP dans une disquette verrouillée en lecture. La documentation qui l'accompagne permet de faire correctement les opérations nécessaires, ainsi, par exemple SCAN ayant détecté la version B du virus Ping-Pong sur la disquette B:, pour lancer CLEAN-UP il suffit de faire : « CLEAN B: [Ping] » puis « ENTER » Après, une désinfection toujours re-utiliser SCAN pour s'assurer que le travail a été fait complètement, parfois il faut deux passages.

INCOMPATIBILITES : à proprement parler aucune, bien que cela soit à nuancer. SCAN ne fait que détecter les virus, bien qu'une option permette d'effacer la partie du fichier où il se trouve, éviter cette option incomplète de désinfection. Donc avec SCAN aucun problème, si ce n'est le doute d'une présence de virus, et il faut donc protéger au moins son disque dur, le plus simple est alors d'utiliser LIFEGUARD avec l'option /P, elle interdit toute écriture sur le ou les disques durs.CLEAN va agir en effaçant, suivant une méthode sûre expliquée dans la notice, la partie virale. Si le virus se trouvait dans un fichier exécutable, il y a un risque de voir ensuite ce fichier ne pas fonctionner correctement, mais ce n'est pas toujours le cas. De toute façon le virus ayant été enlevé, on peut toujours restaurer le fichier en question en reprenant sa disquette originale.

CONSEIL : après une désinfection de virus, refaire un boot, et passer son disque dur (ou sa disquette système) à l'épreuve de SCAN, l'expérience apprend à être prudent. Entre temps, bien éteindre la machine, complètement, le virus pouvant s'être mis de façon transitoire dans la mémoire vive, n'attendant que le moment propice où, abaissant les protections, il puisse se cacher sur le disque.

AUTRES CONSEILS : ne pas faire la collection de virus, surveiller en particulier les disquettes de jeux, que les enfants dès leur plus jeune âge s'échangent, puis ensuite essayent sur le micro de papa. La protection maximale par LIFEGUARD s'impose alors.

MENTION : TRES BIEN logiciels indispensables.

CONCLUSION

J'ai essayé dans les lignes précédentes, de montrer tout l'aspect intéressant, utile, à moindre coût, des partagiciels, qui au niveau de ce qui a été présenté forment un ensemble parfaitement cohérent, et d'une grande fonctionnalité. Dans les semaines à venir je compte compléter cet ensemble par deux nouveaux partagiciels : un pour lister sur imprimante des textes de façon la plus conviviale possible, et un autre pour faire des sauvegardes (BACKUP) moins ennuyeux que celui du DOS. A mon avis, il est préférable de disposer, de petits logiciels adaptés plutôt que d'une grosse usine à gaz. Chaque logiciel ayant une fonction précise, dans le cadre de ce que l'on fait sur son micro. Eviter au maximum les résidents qui encombrent la mémoire et déclenchent parfois des conflits sans crier gare. Ainsi, on peut évoluer soit avec les nouvelles versions, soit par des remplacements le jour où l'on en trouve un, qui pour une fonction donnée convient mieux. Certains patagiciels sont gratuits. Pour les autres, payer sa contribution (toujours minime) tient de l'engagement moral, valorise les partagiciels, permet leur amélioration, encourage l'auteur et, au bout du compte l'utilisateur en profite avec l'amélioration apportée. Les effets vont même beaucoup plus loin et inciteront sans aucun doute les auteurs et diffuseurs de produits commerciaux à une meilleure qualité, pour un coût plus réaliste.

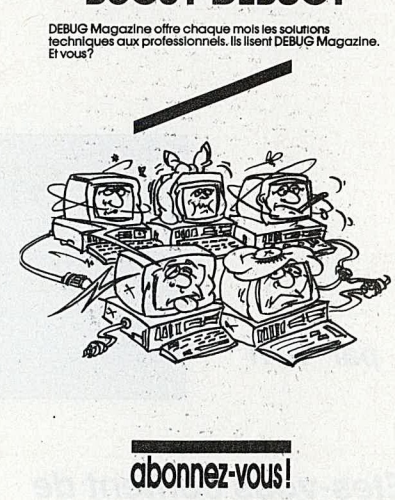
Il est important de signaler aussi la remarquable action développée, pour le développement du SHAREWARE et du FREEWARE par l'association OUF plusieurs fois ici citée. Si l'on n'a pas de modem pour aller chercher les partagiciels sur les serveurs, OUF et son catalogue remarquablement complet, sous forme de disquette, permet à chacun d'accéder à ces immenses ressources et de découvrir une sélection des plus récentes nouveautés mondiales grâce à "La petite chronique des Partagiciels", parution mensuelle sur la lettre de O.U.F.! (envoi d'un exemplaire sur simple demande au siège de O.U.F.!) et sur la « lettre hypertexte », version informatique qui se trouve sur la première disquette mensuelle de la série disquette mensuelle de la série « La petite chronique des partagiciels ».

Ces pages sont offertes gracieusement par Debug Magazine à l'Association OUF. Le choix des textes et la rédaction sont assurés par les membres de OUF, sans aucune intervention de Debug Magazine.

Pour tous renseignements, contactez les à l'adresse suivante :

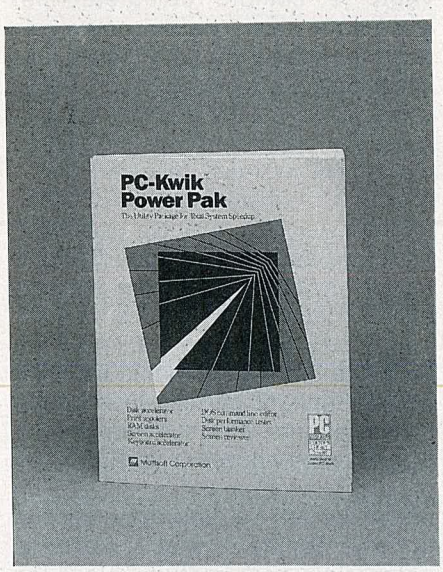
27, rue des Bluets, 75011 Paris.

Tél : (1) 43 38 02 58
Fax : (1) 43 38 08 73



OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF - OUF -

PODIUM



Si vous êtes pressés, arrêtez-vous chez Multisoft

*par *ART**

Etes-vous content de la vitesse de votre ordinateur ? Jamais encore, je n'ai obtenu de réponse positive à cette question. (La course contre la montre ne cesse jamais, et c'est extrêmement coûteux). Généralement, pour aller plus vite, vous achetez un nouvel ordinateur : un 386, un 486 ou, pourquoi pas, bientôt un 586, selon finances et les cadences d'Intel.

Dans cette nouvelle rubrique, nous allons vous présenter des matériels ou logiciels que nous avons particulièrement aimé, pour leur utilité, originalité ou qualité. Si vous appréciez particulièrement un produit, soumettez-nous sa candidature. Les bons produits méritent d' être connus!

Il y a cependant d'autres moyens pour améliorer la situation, moins chers et aussi efficaces, sans remplacer tout de suite la totalité de votre matériel. Parmi les moyens les plus efficaces, on remarque l'ensemble des utilitaires PC-KWIK de Multisoft Corp.

Dans un petit ensemble (un manuel et une disquette), vous achetez de la puissance pure, un jeu de cinq programmes qui accélèrent le fonctionnement de votre système. Plus précisément, PC-KWIK supprime certains goulets d'étranglement qui sont les principales sources de ralentissement de votre système.

En fait, un ordinateur n'est pas lent uniquement à cause la vitesse de l'horloge de son processeur. Il est fort probable qu'actuellement vous n'utilisez la puissance de votre processeur qu'à 10 ou 20 %. La plupart du temps il sommeille, en attendant que vos doigts frappent les touches, que la carte vidéo termine l'affichage d'un écran ou que la tête du disque dur se positionne

[SERVICE LECTEURS : INTERESSANT = 28 / MOYEN = 29/ MAUVAIS = 3 0]

sur la piste ou le secteur indiqué. En vérité, ce qui est le plus lent dans un ordinateur, ce sont des périphériques : mémoire centrale, bus, écran vidéo, imprimante et le disque. Sans le savoir, vous avez peut-être l'ordinateur beaucoup plus performant que celui que vous observez - il suffit de lui donner quelques « coups de pouce ».

ACCELERATEUR D'IMPRIMANTE

Commençons par l'élément le plus lent. Une imprimante typique imprime à 80 à 200 cps (caractères par seconde). Le processus est ralenti par des retours de tête, avancement du papier, etc...; typiquement il faut diviser ces chiffres par deux pour obtenir le flux réel. En résultat, la moindre impression verrouille définitivement le système. Résigné, vous n'avez qu'à aller prendre un café et attendre que ça se termine. Le spooler de PC-KWIK fonctionnant en arrière-plan, intercepte à la vitesse de la mémoire centrale les données destinées pour l'impression, libère instantanément l'application et les vides vers l'imprimante au fur et à mesure de l'avancement du papier. Par rapport à la fonction PRINT livrée avec MS DOS, son avantage est sa petite taille (seulement le programme lui-même de 17Ko réside en mémoire additionnelle (étendue ou expansée) et surtout, le partage automatique de la mémoire allouée avec SUPER PC-KWIK, l'accélérateur des disques. En fait, vous n'avez pas à vous préoccuper par la quantité de la mémoire réservée aux tampons du spooler : si vous avez alloué à SUPER PC-KWIK 2Mo de la mémoire étendue, lors de l'impression le spooler va automatiquement y « prélever » la quantité nécessaire pour contenir le texte à imprimer et la libérer ensuite au profit de SUPER PC-KWKIK au fur et à mesure de l'impression. Une petite fenêtre de contrôle peut apparaître sur l'écran pour piloter l'impression : la suspendre, vider les tampons, envoyer des séquences des caractères de contrôle, l'arrêter pour changer le papier, réinitialiser l'imprimante et modifier les paramètres de transmission. Un algorithme de compression permet d'économiser la quantité de la mémoire utilisée : le taux de compression peut descendre à 90 %.

DISQUE RAPIDE VIRTUEL

Le deuxième utilitaire, RAM DISK, se charge au démarrage à partir du COMMAND.COM et partage aussi la mémoire tampon avec SUPER PC-KWIK. Il est paramètrable de 4Ko à 16Mo et on peut y créer de 16 à 2048 fichiers. Sa particularité est qu'il n'utilise réellement que la quantité de la mémoire physiquement occupée par les fichiers qui y sont stockés. Par rapport aux disques virtuels « ordinaires » (et notamment VDISK livré avec MS DOS), le partage dynamique de la mémoire a cependant un (relatif) défaut : on ne peut pas « récupérer » des fichiers « effacés » : en cas d'effacement d'un fichier, la mémoire est automatiquement restituée au PC-KWIK. L'avantage de cette méthode est que l'on ne gaspille pas la mémoire. J'utilise couramment ce disque virtuel pour contenir COMMAND.COM (en modifiant en conséquence la variable COMSPEC=d :COMMAND. COM, D: étant le disque virtuel) - cela accélère sensiblement le chargement des programmes lors de retour au DOS.

ACCELERATEUR DU CLAVIER

L'élèment suivant, c'est l'accélérateur du clavier. Chargé en mémoire, il n' utilise que 8.5Ko. A la différence d'autres utilitaires globalement tous les problèmes liés au clavier, sans se limiter seulement à l'accélération de la vitesse de la répétition. Dans des systèmes standards, la vitesse de répétition des touches est fixée à 8 caractères par seconde, avec un délai d'une demi-seconde entre les caractères. PC-KWIK peut accélérer la vitesse jusqu'à 100 cps - le curseur peut traverser l'écran plus vite qu'un éclair. Cette accélération est progressive, lente au départ, en augmentant avec le temps d'appui d'un doigt sur la touche. Un autre paramètre ergonomique est pris aussi en compte lors de l'accélération verticale : elle est deux fois moins rapide que l'accélération horizontale. En plus, il vous permet de désigner les touches à accélérer plus que les autres. La capacité du tampon préemptif du clavier est aussi pris en compte, on ne risque pas des débordements. L'avantage principal réside dans un éditeur des commandes du DOS : le programme mémorise toutes les commandes que vous donnez au DOS, affiche dans une fenêtre leur historique et vous permet de les évoquer à nouveau, après une modification éventuelle. Un véritable gain de temps, indispensable au bout de quelques jours d'utilisation !

ACCELERATEUR DE L'ECRAN

L'accélérateur d'écran est l'élément le plus « visible » de l'ensemble, produisant l'effet que vous allez immédiatement remarquer grâce à la déviation des fonctions d'affichage du DOS et BIOS. Après son installation (environ 7Ko de la mémoire), le défilement de vos écrans va accélérer de deux à quatre fois. Il est compatible avec tout type de l'adaptateur vidéo : MDA, Hercules, CGA, EGA et VGA. Et si l'accélération s'était révélée trop efficace (par exemple, vous ne pouvez plus rien remarque sur l'écran, tellement c'est rapide), des combinaisons des touches permettent de contrôler sa vitesse d'affichage. Il apporte aussi un noircisseur de l'écran, paramètrable de 1 à 60 minutes, pas très original car dépourvu des fioritures graphiques. Son trait unique réside cependant dans une mémorisation d'écrans : tous les affichages qui défilent sont mémorisé dans un tampon (partagé dynamiquement avec PC SUPER KWIK) et peuvent être rappelés pour une visualisation ultérieure. La taille de ce tampon est paramètrable et peut atteindre la totalité alloué au PC SUPER-KWIK.

ACCELERATEUR DES DISQUES

Enfin, SUPER PC-KWIK : un accélérateur des disques, ou une « mémoire cache ». En fait, comme nous l'avons vu précédemment, c'est d'abord un gestionnaire de la mémoire nécessaire pour les autres programmes de l'ensemble. Son code occupe quelques 80Ko dans la mémoire conventionnelle (en plus du tampon éventuel que vous y allouez). Cependant, si vous avez de la mémoire LIM, cette taille peut être réduite à 10Ko car une partie du code va se reloger.

PC SUPER-KWIK s'intercale entre le programme et le disque physique, pour mémoriser dans un tampon en mémoire vide les secteurs (ou pistes) récemment utilisées. Cela permet de réduire le nombre des accès physiques au disque (dur ou disquette) et améliorer remarquablement les temps de réponse du système. Le facteur d'accélération dépend notamment de la taille de la mémoire que vous allouez à SUPER PC-KWIK, de type de l'application, de fragmentation du disque et des performances de la machine. Un cache de 64Ko améliore déjà sensiblement des performances. Un plus large, de 128Ko, va dramatiquement changer les performances de votre système. Même les systèmes les plus sophistiqués tirent bénéfice de PC SUPER-KWIK. Lors de nos tests d'un ordinateur POLYWELL 486/33 MHz, équipé

d'un disque 15ms connecté sur un contrôleur caché ESDI, le programme de test sous FoxBase, manipulant extensivement une grande base de données de 20Mo (multiples recopies physiques, réindexations et remplacements) s'exécutait 2 fois plus vite avec PC SUPER-KWIK installé (120 min au lieu de 210 min). Pourquoi ?

La réponse est simple : un processeur 486 à 33 MHz peut être des dizaines de fois plus efficace qu'un processeur lent. Par contre, le meilleur disque dur n'est que deux ou trois fois plus rapide que celui d'un XT. Puisque l'on accède aux données à partir du cache de la mémoire centrale, le résultat est plus sensible avec un processeur rapide.

L'action du cache est principalement destinée à améliorer des accès en lecture. Les écritures dans des fichiers doivent être immédiatement répercutés sur le disque physique pour une évidente raison de sécurité - en cas d'une coupure de courant, vous risquez de perdre tout ce qu'il y a dans le cache. Multisoft a cependant développé des algorithmes spéciaux pour optimiser tant des accès en lecture qu'en écriture. PC SUPER-KWIK est souple, s'adpate à tout type de mémoire, tout disque et contrôleur (MFM, RLL, IDE, ESDI et SCSI). Cependant, pour tirer le meilleur profit de ses possibilités, une expérimentation est nécessaire : vous avez intérêt à changer certains paramètres, tester différentes options (notamment le choix d'algorithme d'optimisation), pour trouver la combinaison qui correspond le mieux à votre système et type des éléments installés.

SAUVETAGE D'UN BULL

J'ai testé l'effet d'installation d'un PC-KWIK sur un « vieux » PC de BULL. La machnine, utilisée pour une simple gestion d'un fichier d'adresses gisait abandonnéd dans un coin prenant poussière - sa lenteur rebutait tout utilisateur potentiel. J'ai commencé par le reformatage total du disque, tout en changeant le facteur d'interlacement des secteurs (l'initial donnait le taux de transfert 4 fois inférieur à la capacité potentielle du contrôleur !). Ensuite, j'ai installé un DR DOS 5, PC-KWIK et Norton Commander. L'ancienne dBase III a été remplacée par une FoxPRO. L'application originale n'a pas été touchée (à part la correction de quelques bugs). Par rapport à l'état initial, les temps d'une réindexation du fichier, d'une recherche ou d'une sélection d'adresses ont été divisées par plus de dix. Les taches qui duraient 2 ou 3 heures, s'exécutaient en 15 minutes ! Le résultat a été éclatant : la personne chargée du fichier a abandonné le fichier manuel et est revenue à la machine.

En conclusion, si votre ordinateur donne des signes de fatigue, si vous commencez à avoir assez de l'attendre, si vous commencez à avoir des complexes voyant les nouvelles bécanes de vos collègues, avant de jeter votre matériel, essayer PC-KWIK. Le résultat va vous surprendre !

PRODUITS TESTES

PC KWIK Power Pak
version 1.59
prix : $ 129.95 (environ 715 Frs.)

PC KWIK Power Disk
version 1.04
prix $ 79.95 (environ 440 Frs.)

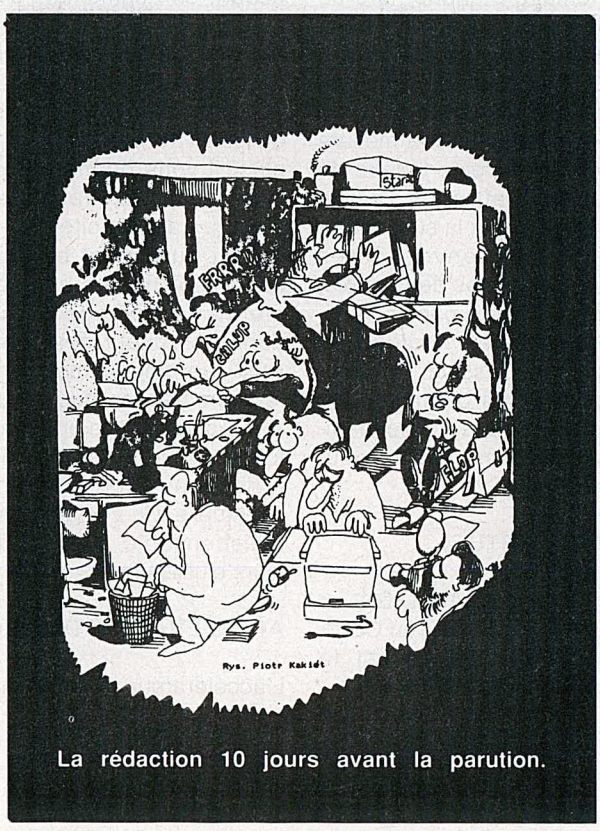
Multisoft Corp.
15100 S.W. Koll Pkwy
Suite L
Beaverton, OR 97006
USA
Tél : (503) 644 5644
Fax : (503) 646 8267
Compuserve 76004, 151

Multisoft édite aussi un autre utilitaire, appelé Power Disk.
C'est essentiellement un défragmenteur et l'outil de test de la surface.

Power Disk peut être lancé automatiquement, par une ligne de commande ou piloté à la main, par des menus et touches de fonction.

Le vérificateur teste la surface magnétique du disque et verrouille les mauvais secteurs. C'est le plus rapide, vérificateurs que je connais sur le marché, il travaille par des grappes des secteurs (clusters) et non à un, comme les autres. L'optimiseur dispose des options de choix de stratégie (on peut choisir l'emplacement des fichiers nominativement ou en indiquant les Attributs) et choix de niveau de sécurité (sans protection ou avec, en comparant les écritures aux lectures). Nous ne pouvions pas le tester sous DR DOS, à cause d'une incompatibilité interne d'un appel à une fonction du DOS, à laquelle Multisoft promet d'apporter une solution.

D'une présentation rustique, Power Disk est utile et efficace. Il complète utilement la panoplie des outils dans la mallette d'un technicien.



La rédaction 10 jours avant la parution.

IDEE DU JOUR : Documentor

par G. Korwin

Dans cette rubrique, intitulée « IDEE », nous publions les idées diverses sur les programmes ou dispositifs à réaliser.
Des idées géniales, imprévisibles, nouvelles, créatives, farfelues (qui sait?) en somme, la diversité des idées à réaliser.

En effet, DEBUG MAGAZINE manque (comme tous les autres, par le temps qui court) de pétrole, mais pas d' idées.
Chaque programmeur se pose quotidiennement la question : que faire lorsque je terminerai le programme en cours ?
Ce programme n'est jamais terminé, mais à des moments rompus, on commence une nouvelle réalisation...

Tous, nous avons aussi des besoins. Tous les jours, nous nous disons : « ah, si cela existait, je serai moins ennuyé »!

Décrivez-nous vos besoins, envoyez les idées, même celles qui semblent le plus bizarre. Qui sait ? Peut-être allez vous déclencher une découverte révolutionnaire ?

L'idée de DOCUMENTOR m'est venue suite à une longue et douloureuse expérience de développeur, qui consacre tout son temps, compétences et passion au développement du programme et souffre quand on lui demande, en plus, une documentation écrite - le manuel.

En effet, vous vous enthousiasmez pour un projet, une idée à réaliser, vous commencez la programmation, vous alignez les instructions du code, vous déboguez, vous testez, améliorez, perfectionnez, optimisez, parfois même vous réecrivez tout à zéro - c'est monnaie courante dans notre profession.

Arrive enfin le jour de la livraison : fiers de votre enfant nouveau-né, vous le présentez au client. Généralement, il est content qu'on lui ait enfin résolu son problème. Il suit avec vous la présentation et les manipulations d'utilisation, sans tout comprendre, généralement vous remercie. Mais infailliblement, à la fin, tombe la question fatidique : « et où est le manuel? ». Évidemment, vous avez dépensé toute votre énergie et tout votre temps à concevoir le meilleur programme possible, mais de là, écrire en plus un livre... c'est une autre affaire.

Il est toujours possible de créer un petit document de manipulation, indiquant grosso-modo ce qu'il faut faire pour démarrer les fonctions essentielles. Mais ce document n'est pas un manuel exhaustif, listant en détails toutes les fonctions, formes acceptées à la saisie, transformations subies par les données, résultats des processus que vous avez installé à l'intérieur du programme et dont vous êtes souvent le seul à connaître l'existence (dans des programmes de mes collègues il m'est arrivé de découvrir certaines fonctions existant depuis le début mais, faute d'une quelconque documentation, totalement inconnues!).

En fait, le travail de création d'un bon manuel est au moins aussi long et fastidieux que la création du programme lui- même.

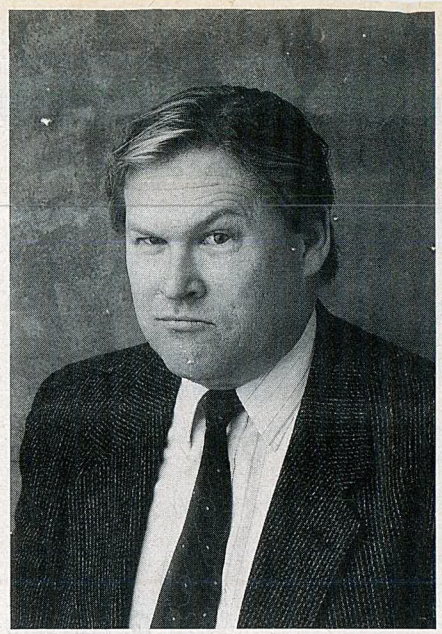
Et pourtant, le programme lui même est déjà la meilleure documentation qui peut exister! Le problème vient du fait que le langage utilisé pour l'écrire est généralement totalement obscur pour des utilisateurs ordinaires. Ce qui est dramatique c'est qu'on ne peut pas demander, par exemple, à un comptable, de comprendre un programme d'amortissement à partir du code source! Le commun des mortels, simplement, ne comprend pas ni C, ni Pascal, ni xBase.

Le travail de programmation consiste en traduction du langage utilisé par le commun des mortels en langage compréhensible par les machines. En réalité, la documentation existe, mais elle n'est compréhensible que par quelques rares élus, le langage utilisé communément par les autres n'est pas le même.

Cette simple constatation m'amène à proposer l'idée du mois : DOCUMENTOR, un programme de documentation automatique qui créerait la documentation à partir du code source. DOCUMENTOR doit être capable de lire, par exemple, du code en Pascal et générer un texte (ou au moins une proposition, un squelette, une maquette pour le Traitement de Texte), qui pourrait être utilisé pour écrire rapidement un manuel de référence. En fait, DOCUMENTOR ferait le travail inverse à la programmation : il traduirait en code humain le langage utilisé par le commun des ordinateurs. Il doit lister les données acceptées par le programme, les cheminements des menus et le fonctionnement des différentes procédures (fonctions) accessibles par l'utilisateur et les résultats obtenus.

Quelques tentatives similaires existent déjà : il y a quelque programmes de référencement de variables pour des langages courants. MULTILOG était pourvu d'un utilitaire de documentation automatique. Mais un programme de documentation, pour les langages de programmation les plus répandus, tel qu'il est présenté ci-dessus, n'existe pas. Et pourtant, il nous simplifierait grandement la vie. Alors, à vos claviers...

John C. Dvorak



L'HISTOIRE DES LOGICIELS RATES

Par les temps qui courent, il y a beaucoup de déchets disponibles pour les micro-ordinateurs. Par déchets je comprends des logiciels sans aucune utilité.

Il semble qu'il y ait trois catégories principales de ces logiciels inutiles, visibles de loin. Ce qu'ils ont en commun, c'est que toutes les entreprises qui les fabriquent espèrent faire de l'argent sur le même marché - le marché de l'utilisateur naïf, sans formation, ni expérience. C'est comme si l'on construisait des voitures pour des personnes qui ne savent pas conduire.

Les premiers produits dans ce groupe sont des soi-disant shell du dos. Ces produits commencent à devenir les principaux sujets de sarcasmes parmi les journalistes des principaux magazines informatiques. Dans les halls d'entrée on peut entendre : « Laisse moi deviner : ce type veut nous parler au sujet de son nouveau shell du DOS! », suivi par un rire sardonique. Les personnes qui peuplent les journaux forment le groupe le plus sensibilisé. C'est parce qu'après avoir vu 482 shell du DOS vous devenez définitivement fatigué.

Si vous l'ignorez encore, un shell du DOS c'est un programme qui recouvre l'écran normal, tel que vous avez l'habitude de le voir quand vous utilisez un ordinateur de type IBM PC. A la place d'un simple C> vous voyez un écran, plein de listes et détails sur le système, l'écran qui est supposé rendre plus aisées vos ennuyeuses tâches quotidiennes. Au lieu d'écrire au prompteur, par exemple, « FOXPRO », vous visualisez une longue suite d'écrans pour pointer finalement le fichier qui contient le programme FOXBASE. Ensuite vous devez frapper la touche ENTREE, une touche de fonction ou une lettre et le programme commence à s'exécuter. Si quelqu'un demande mon opinion, j'ai l'impression qu'il est plus simple juste d'écrire « FOXPRO ».

L'idée du shell part d'un louable désir d'aide. Ses promoteurs espèrent rendre la vie de l'utilisateur plus facile. Pour visualiser les fichiers sur disque, à la place de mémoriser la très compliquée commande « DIR », vous n'avez qu'à pointer quelque chose dans le shell et il va lister le répertoire pour vous. Par sa vraie nature, le shell du DOS présume que vous êtes une personne complètement bête et stupide, sans aucun espoir. Je suppose que les shells sont d'une certaine valeur pour des idiots originaux qu'on peut rencontrer ici et là. Sans rancune pour les utilisateurs intelligents qui aiment ces choses. Amen.

Enfin, un nouveau shell arrive toutes les semaines. J'en reçois certains d'entre eux avec une note attachée qui dit généralement « John, je sais que tu n'aimes pas les shells du DOS mais je trouve que ce XYZ est vraiment différent ... bla, bla, bla ». Ils n'arrivent vraiment pas à comprendre.

Le produit concurrent est l'utilitaire d'aide en-ligne. Ces programmes sont d'une certaine utilité aux amateurs, pendant un laps de temps, disons - une semaine. Essentiellement, ce sont les manuels du DOS sur la disquette. Quand vous avez un problème, vous écrivez juste « AIDE » (ou plutôt « HELP », ce qui n'est pas d'une grande aide - NDLR) ou, universellement, vous frappez la touche F1. Ensuite, vous devez vous promener dans une longue série de menus, lectures et explications. En somme, vous apprenez comment écrire « DIR » pour voir le répertoire. La meilleure méthode pour accomplir cette difficile tâche d'apprentissage est de demander à un ami de vous montrer toutes les ficelles - c'est rapide et efficace.

La dernière mouture de ces programmes utilise une sombre appellation « l'aide contextuelle ». Cela veut dire que le programme tient compte de la position du curseur, regarde le mot concerné et essaye de trouver le fichier d'aide en relation. Si vous pointez le mot « SORTIE » et déclenchez le programme, il explique (peut-être) ce que « SORTIE » veut dire.

Finalement, il y a des produits que j'appelle les générateurs de menus et d'annotation. Les générateurs de menus sont crées pour des types de services informatiques qui sont supposés créer un nombre restrictif des applications pour d'autres services. La présomption de ces logiciels (et du service informatique) est que les utilisateurs ne vont utiliser qu'un nombre limité de quelques applications spécifiques, point terminal.

Un typique menu « approuvé » apparaît quand la machine démarre et propose : « VOULEZ VOUS FAIRE — A) Word — B) Lotus, —C) imprimer le répertoire de fichiers — D) TERMINER ». Pour cet éprouvant travail le type de l'informatique empoche 350.000 Frs par an.

Encore pire, les générateurs de menus sont des logiciels d'annotation. Ces programmes sont une combinaison du plus mauvais aspect des programmes d'aide et des générateurs des menus. Ils encouragent quelques personnes mal intentionnées à ajouter les écrans d'aide « personnalisée », qui vont apparaître désespérément à tout moment et à chaque endroit. Ces écrans sont généralement crées par le type du service informatique ou un ingénieur informaticien, qui ne savent autrement rien écrire. En d'autres mots, au lieu d'acheter les programmes d'aide mentionnés plus haut, l'informaticien écrit ses propres fichiers d'aide, en croyant qu'il peut faire mieux. Je suppose que cela a pour seul but de meubler son curriculum vite.

Un jour, un opérateur ténébreux va combiner toutes ces idées en un grand programme - l'Universel Générateur Paramétrable des Menus d'Aide Contextuelle (UGP MAC DOS, pour abréger). Je parie qu'il va autoriser à l'utilisateur qu'une seule chose : lister le répertoire!

Mon conseil à l'utilisateur timoré : achètez un livre et apprenez comment utiliser correctement ces machines!

[SERVICE LECTEURS : INTERESSANT = 34 / MOYEN = 35 / MAUVAIS = 36]

PETITES ANNONCES LECTEURS

Je vends, cause départ, 4 logiciels originaux, neufs & complets, pour Apple Macintosh, à la moitié de leur prix : Adobe Illustrator (2 950 F), Quark XPress (3 750 F), Aldus Page Maker (4 050 F), Aldus FreeHand (2 975 F). Tél : (16) (1) 47 83 58 89.

Vends **IBM PS2 8550 DD** 60 Mo : 6 000 F.
Carte fax 9600 GP3 + Modem 2400 + Répondeur, possi-
bilité de scanner : 4 000 F.
Carte 386 25 MH2, 64 Ko cache 4 Mo RAM : 4 000 F.

Vends **Logiciel Picsou** (gestion de vos comptes et de votre budget familial). Contribution pour l'auteur : 300 F. Modry Rémy - 20, rue du Ballon - 68260 - Kingersheim. Tél : 89 57 29 54 après 20 Heures.

Ordinateur AMSTRAD PC 2286/40 (286/12 MHZ, disque dur 40Mo), écran couleur 14 HRCD (piqué 0,29) lecteurs 5'25 et 3'5, nombreux logiciels : 6 000Frs. Michel Barraud, bureau : 42 92 46 36, domicile : 45 47 28 28.

VOUS ETES DYNAMIQUE

Debug Magazine recherche pour son service publicité

ATTACHE (E) COMMERCIAL (E)

* Connaissance de l'industrie informatique souhaitée.
* Expérience des outils bureautiques (traitement de texte, bases de données, tableurs) nécessaire.

REMUNERATION MOTIVANTE EN FONCTION DES RESULTATS

Pour prendre rendez-vous, envoyez un Curriculum Vitae.

Pour les abonnés de DEBUG Magazine, les petites annonces sont gratuites (joindre la bande témoin du dernier numéro reçu). Pour les autres lecteurs, 50 f, par coupon renvoyé. Règlement par chèque ou carte bancaire à l'ordre de: DEBUG Magazine - 3, rue de l'Atlas - 75019 Paris.

J'ACHETE

J'ECHANGE

JE VENDS

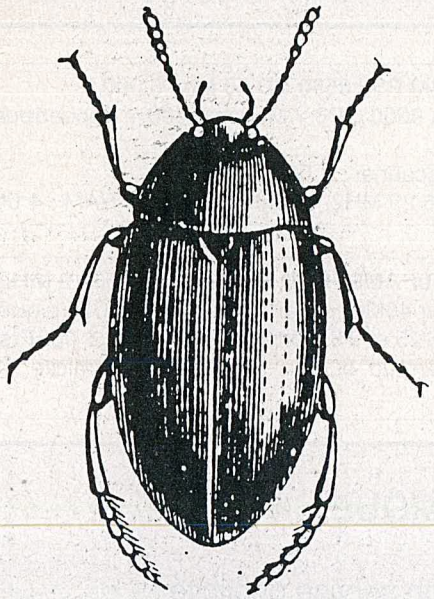
DEBUG MAGAZINE

BULLETIN DE PETITES ANNONCES

☐ J'ÉCHANGE ☐ J'ACHÈTE ☐ JE VENDS

☐ Je suis abonné à DEBUG MAGAZINE : ci-joint bande témoin du dernier numéro

☐ Je ne suis pas abonné à DEBUG MAGAZINE; ci-joint un chèque de 50



SONDAGE PERMANENT

Nos lecteurs ont attribué les scores suivants aux articles parus dans DEBUG MAGAZINE n° 10:

1° PROTECT par Bruno Vasseur

2° Procédures de sortie en Turbo Pascal par Serge Grun et TurboC++ et Zortech C++ par Serge Grun

3° OUF par Georges Bourne

4° PODIUM par *ART*

Nous félicitons les auteurs!

LE PROCHAIN NUMERO

Debug Magazine n° 12 *(mai 1991)*

Parution 25 avril 1991

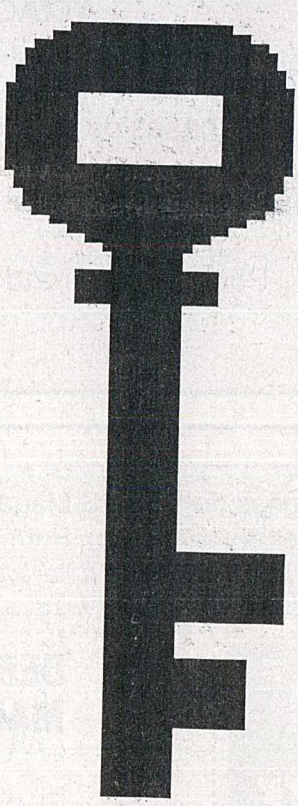
Sujet : OUTILS SYSTEME DU DOS

Cahier Dr Dobb's Journal :
- Comment visualiser l'allocation mémoire sous DOS
- Un accélérateur cache pour disques durs en C
- Comment écrire des (DVD) pilotes MS DOS en C

Et Aussi :
- « Programmation Window» (3me partie) par S Grun
- « Les structures évoluées de données » par F Berton
- « Génération automatique de grilles pour mots cases » par L Chambaret
- « Des arbres programmatiques pour une analyse naturelle » par J Ch Riat

Evaluation : Paradox
Podium : Ordinateurs Polywell
Idée : ??? !
Et comme d'habitude : page de **John C Dvorak**

Page de John C Dvorak

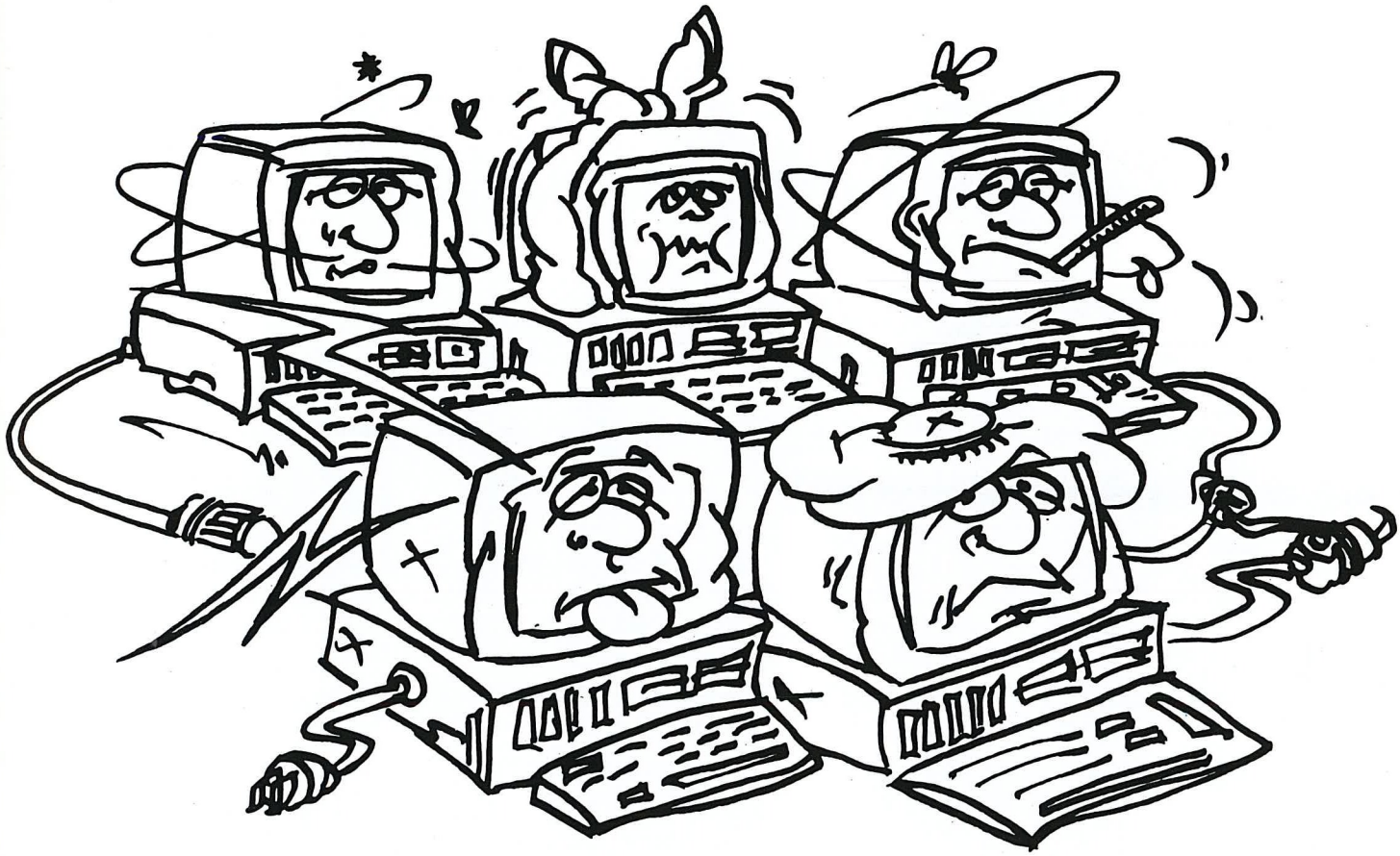


DEBUG MAGAZINE vous invite à faire partager votre expérience avec d'autres lecteurs. Nous sommes intéressés par la publication de textes choisis consacrés à la programmation ou à la technologie en général. Nous pouvons publier vos textes qui apportent une information pertinente à d'autres lecteurs. Votre travail sera rémunéré - demandez notre Dossier Collaborateur.

DEBUG MAGAZINE s'intéresse à vous. Nous aimerions recevoir des informations sur les nouvelles technologies, les nouveaux produits ou les développements de votre entreprise. Envoyez-nous vos communiqués de presse, par courrier ou par télécopie. Et si vous voulez donner un coup de projecteur sur une info (ou une vacherie fondée à dire), ne vous gênez pas.

BUGS? DEBUG!

DEBUG Magazine offre chaque mois les solutions techniques aux professionnels. Ils lisent DEBUG Magazine. Et vous?



abonnez-vous!

Backup PRO

rapide, facile et fiable

THE SOFTWARE TOOLWORKS

- 10 Mo en 4 minutes
- jusqu'à 90% de compression de données
- nouvelle interface graphique en couleurs
- mode menu et commande
- vérification et/ou correction automatique d'erreurs
- compatible avec toute unité du DOS et Réseaux Locaux

Le meilleur logiciel de sauvegarde du monde!

OFFRE SPÉCIALE EN CADEAU A CHAQUE ABONNÉ !